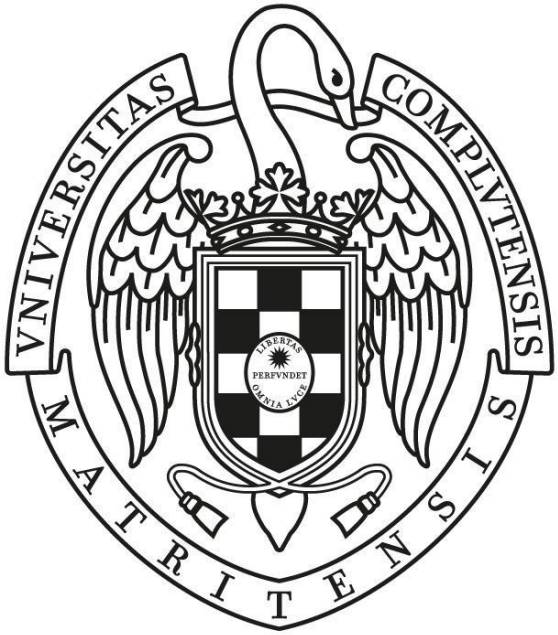
# Autenticación y verificación de usuarios mediante dinámica del tecleo




Fernando Rivilla Bravo

*Dirigido por*
Enrique Martín Martín

Trabajo de fin de Grado en Ingeniería del Software

Facultad de Informática
Universidad Complutense de Madrid
Curso 2016-2017

15 de junio de 2017

# Agradecimientos

En primer lugar, agradecer a Enrique Martín Martín, mi tutor, toda su ayuda, consejos y colaboración.

A mis amigos, por acompañarme en esta experiencia inolvidable. Porque, hagas lo que hagas en la vida, no es legendario si tus amigos no están ahí para verlo.

A mis padres, Miguel y Chelo, por su apoyo incondicional 365 días al año, que me ha dado infinitas fuerzas para seguir siempre hacia delante y ser capaz de conseguir todas mis metas.

A mi hermano Daniel, por ser mi mejor amigo, por aguantarme hasta cuando es imposible y por estar siempre a mi lado contra viento y marea.

Y a ti, Irene, por ser la luz que alumbra mi vida, que camina a mi lado por este duro camino y que hace de mi la persona más feliz del mundo entero.

# Resumen

Dentro del área de la autenticación biométrica, uno de los campos que ha suscitado mayor interés en los últimos años ha sido la dinámica del tecleo. En él, se estudian multitud de técnicas de clasificación de usuarios con el objetivo de encontrar un sistema de autenticación alternativo a las contraseñas utilizadas en la actualidad. Todas ellas se basan en las diferentes características biométricas que las personas mostramos al utilizar un teclado informático. Por tanto, se propone realizar un estudio de distintas técnicas de clasificación de usuarios mediante dinámica del tecleo e implementar un sistema que utilice algunas de ellas.

# Palabras clave

Biometría, Dinámica del tecleo, Clasificador, Contraseña, Seguridad

# Abstract

Within the area of biometric authentication, one of the fields that has created most interest in recent years has been the keystroke dynamics. In it, we study a multitude of user classification techniques in order to find an alternative authentication system to passwords used today. All of them are based on the different biometric characteristics that people show when using a computer keyboard. Therefore, we propose to perform a study of different user classification techniques using keystroke dynamics and integrate them into a system in order to evaluate their accurracy.

# Keywords

Biometrics, Keystroke dynamics, Classifier, Password, Security.

# Índice

Anexos:

# 1. Introducción

En los últimos años, la validez de los métodos tradicionales de seguridad, como son las contraseñas, se ha visto cuestionada por los múltiples robos o pérdidas de datos. Ante estos problemas, surge la necesidad de encontrar nuevos sistemas de control más eficaces que garanticen lo más posible nuestra seguridad [1].

Esto nos ha llevado a fijarnos en otros campos de estudio como la biometría. La biometría es la aplicación de técnicas matemáticas y estadísticas sobre los rasgos físicos o de conducta de individuos, para verificar su identidad [2]. Las huellas dactilares, la retina, el iris o los patrones faciales representan ejemplos de características físicas (estáticas), mientras que entre los ejemplos de características del comportamiento se incluye la firma, el paso o el tecleo (dinámicas). A raíz de esto, hemos decidido hacer un estudio sobre esta última característica, el tecleo o dinámica de pulsación, buscando un método seguro de identificación a través del teclado de un terminal.

La dinámica de pulsación evalúa patrones y ritmos de escritura habituales de una persona mientras escribe en el teclado de un dispositivo informático. Estos patrones son característicos de una persona y permiten distinguirla de las demás, al igual que la escritura o la firma [3]. Además, esta técnica tiene múltiples ventajas: no intrusiva, bajo coste, fácil de usar, no requiere de soporte hardware, continua (podemos recopilar los datos de un usuario mientras trabaja incluso sin su cooperación o conciencia) entre otras [1,3].

En este trabajo, nos vamos a centrar en el estudio de varias técnicas estadísticas de clasificación que más tarde nos servirán para la creación de un sistema de dinámica del tecleo. A través de él, podremos identificar usuarios según sus características biométricas y analizaremos un conjunto de muestras de diferentes individuos con el fin de ver que técnica es la más efectiva y qué características son mejores para la identificación de individuos. El trabajo realizado junto con todos los resultados pueden verse en el siguiente repositorio de GitHub:

https://github.com/ferrivi14/TFG-Keystroke_Dynamics

## 1.1. Antecedentes

El estudio de la biometría de pulsación empezó a finales de los años 70 [1]. Al principio, se centró en el uso de textos estáticos y predeterminados, como contraseñas. Pero a medida que fue avanzando, poco a poco, se empezó a utilizar el texto libre (da igual lo que escriba el usuario mientras que lo escriba a su manera). Esto permitió alcanzar un nuevo tipo de sistema en el que nos identificamos de manera continua durante toda nuestra sesión, no sólo al acceder al sistema.

Este tipo de estudio se ha visto incrementado en los últimos años, favorecido por las nuevas tecnologías y la gran necesidad de mayor seguridad en nuestros dispositivos. Esto ha provocado que surjan nuevas vertientes de estudio [1]:

- *Dinámica de pulsación en dispositivos móviles*: el uso de teclados virtuales, múltiples sensores, predictores de textos y otros detalles que amplían enormemente el conjunto de posibilidades de estudio en el campo de la dinámica de pulsación.
- *Biometría suave*: estudio de ciertas características que permiten diferenciar factores como el género o la raza pero que no tienen carácter distintivo.
- *Dinámica de pulsaciones para el reconocimiento de emociones*: como la dinámica de pulsación de un usuario está influenciada por su estado mental, ha surgido interés en el estudio de su estado emocional utilizando técnicas avanzadas de aprendizaje máquina.

## 1.2. Objetivo

El objetivo de este trabajo era implementar un sistema de autenticación y verificación de usuarios basado en la dinámica del tecleo. Para ello, era necesario realizar un estudio en profundidad de múltiples técnicas. De aquellas que estudiamos, elegimos un subconjunto, el cual fue incluido en el sistema. Además, queríamos evaluar el rendimiento y calidad de cada una de ellas en diferentes situaciones. Para ello, realizamos una batería de pruebas usando múltiples combinaciones de características y parámetros. La colección de resultados obtenida nos permitió resolver diferentes preguntas que fuimos planteando a lo largo de todo el trabajo. Algunas de ellas son:

- ¿Qué técnica ofrece mejores resultados?
- ¿Qué combinación de características es mejor para cada técnica?
- ¿Cuáles son los mejores resultados para cada combinación de características en una técnica?
- ¿Qué características son mejores independientemente de la técnica?
- ¿Qué parámetros son mejores para cada técnica?

## 1.3. Plan de Estudio

Podemos dividir el desarrollo del proyecto en cuatro partes: Investigación, Obtención de Datos, Implementación y Análisis de Resultados.

### 1.3.1. Investigación

Como punto de partida del proyecto, tuvimos que realizar un estudio previo de diferentes técnicas de clasificación de usuarios mediante distintos artículos de investigación en la materia. En uno de ellos [2], encontramos una comparativa entre los resultados obtenidos en varias investigaciones y decidimos centrarnos en un subconjunto de ellos, que obtenían buenos resultados y utilizaban técnicas, *a priori*, no muy complejas de implementar.

Tras estudiar y analizar este subconjunto de técnicas, decidimos quedarnos con unas pocas, las cuales son las que hemos implementado y a analizado durante el trabajo:

- Detector Manhattan [6]: la distancias entre cada característica de una nueva entrada y las características del vector medio de un sujeto deben ser todas menores que un umbral concreto.
- Media y desviación típica [7]: el sumatorio de las diferencias entre una nueva entrada y el vector medio de un usuario deberá ser menor que la media de las diferencias, calculadas entre el vector medio y otras entradas antiguas; más *n* desviaciones típicas de esas diferencias.
- Z-scores [8]: una nueva entrada debe tener un mínimo de valores dentro de los intervalos de confianza del sujeto a quien dice pertenecer.

### 1.3.2. Obtención de Datos

Un sistema de dinámica de pulsación completo necesita una aplicación de captación de datos y una gran cantidad de entradas por teclado generadas por múltiples personas. Ante la complejidad de crear uno propio desde cero, decidimos centrarnos en el análisis de las diferentes técnicas utilizando conjuntos de datos ya existentes y dejar para un futuro la creación de nuestro propio sistema.

El proceso de encontrar un conjunto de datos fue arduo pero, tras mucho buscar, encontramos un gran conjunto de datos, que disponía de más de veinte mil entradas de 51 usuarios distintos [14]. Cada una de estas entradas registra los tiempos de presión y de vuelo existentes al introducir una contraseña concreta, además del usuario que la realizó, su número de sesión y de repetición.

Este conjunto de datos venia codificado en tres formatos distintos: Excel, Texto plano y CSV. Debido a esto, decidimos hacer una investigación sobre qué tipo de formato nos permitirá extraer los datos de forma más fácil para nuestro proyecto. Tras varias comprobaciones, llegamos a la conclusión de que el formato más adecuado para nosotros era CSV. Decidido esto, implementamos un programa para la extracción de los datos, el cual ordena y clasifica las diferentes características biométricas de cada entrada y las almacena en una estructura de datos.

Sin embargo, antes de comenzar a programar ninguno de las técnicas elegidas, tuvimos que realizar un programa que dividiese nuestro conjunto de datos en dos partes: entrenamiento y test. Esto se debe a que los clasificadores analizan los datos del conjunto de entrenamiento para luego poder comprobar si un usuario del conjunto de test es quien dice ser.

### 1.3.3. Implementación

Antes de implementar cada técnica, realizamos un estudio en profundidad de cada una de ellas para así evitar posibles errores futuros. Hecho esto, comenzamos la implementación de los algoritmos en el siguiente orden:

1. Detector Manhattan ≡ Sistema de autenticación MAN.
2. Media y desviación típica ≡ Sistema de autenticación MEANSTDV.
3. Z-scores ≡ Sistema de autenticación ZSCORE.

Todas ellas siguen un mismo patrón de funcionamiento:

1. Inicialización del algoritmo con sus parámetros específicos (diferentes según la técnica).
2. Entrenamiento.
3. Evaluación o test.
4. Devolución del resultado obtenido.

### 1.3.4. Análisis de Resultados

Una vez completada la implementación de los tres clasificadores, realizamos una batería de pruebas con el conjunto de datos y probando distintas combinaciones de características a considerar y parámetros. Los resultados de cada combinación son almacenados en un fichero para después analizarlos de forma exhaustiva y sacar conclusiones concretas. Este análisis podrán verlo en la sección 5.

# 2. Introduction

In recent years, the validity of traditional security methods, such as passwords, has been questioned by multiple theft or loss of data. Faced with these problems, the need arises to find more effective systems that assure as much as possible our security [1].

This has led us to look at other fields of study such as biometrics. Biometry is the application of mathematical and statistical techniques on the physical or behavioral traits of individuals, to verify their identity [2]. Fingerprints, retina, iris, or facial patterns represent examples of physical (static) characteristics, while examples of behavioral characteristics include signature, step, or click (dynamics). As a result of this, we have decided to carry out a study on this last characteristic, the click or keystroke dynamics, looking for a secure method of identification through the keyboard of a terminal.

Keystroke dynamics evaluate a person's usual writing patterns and rhythms while writing on the keyboard of a computer device. These patterns are characteristic of a person and allow to distinguish it from others, as well as writing or signing [3]. In addition, this technique has many advantages: non-intrusive, low cost, easy to use, does not require hardware support, continuous (we can collect data from a user while working even without their cooperation or awareness).

In this work, we are going to focus on the study of several statistical techniques of classification that later will serve us for the creation of a system of keystroke dynamics. Through it, we will be able to identify users according to their biometric characteristics and we will analyze a set of samples of different individuals in order to detect which technique is the most effective and which characteristics are better for the identification of individuals. The work done with all the results can be found in the following GitHub repository:

https://github.com/ferrivi14/TFG-Keystroke_Dynamics

## 2.1. Background

The study of keystroke biometrics began in the late 1970s [1]. At first, it focused on the use of static and predetermined texts, such as passwords. But as it progressed, little by little, the free text began to be used (no matter what the user writes as he writes in his own way). This enabled a new type of system in which we identify ourselves continuously throughout our session, not only to access the system.

This type of study has been increased in recent years, favored by new technologies and the great need for greater security in our devices. This has led to the emergence of new areas of study [1]:

- *Keystroke dynamic on mobile devices*: using virtual keyboards, multiple sensors, predictive text and other details that greatly expand the set of opportunities for study in the field of keystroke dynamics.

- *Soft Biometrics*: study of certain characteristics that distinguish factors such as gender or race but have no distinctive character.
- *Keystroke dynamics for the recognition of emotions*: as the dynamics of a user's pulsation is influenced by their mental state, interest has arisen in the study of their emotional state using advanced machine learning techniques.

## 2.2. Goals

The main goal of this work was to implement an authentication system and user verification based on keystroke dynamics. For this, an in-depth study of multiple techniques was necessary. From those techniques studied, we chose a subset, which was included in the system. In addition, we wanted to evaluate the performance and quality of each of them in different situations. To do this, we perform a battery of tests using multiple combinations of features and parameters. The collection of results obtained allowed us to answer different questions that we pose over all the work. Some of them are:

- What technique offers the best results?
- What combination of features is best for each technique?
- What are the best results for each combination of features in a technique?
- What features are best regardless of technique?
- What parameters are best for each technique?

## 2.3. Workplan

We can divide the development of the project into four parts: Investigation, Data Collection, Implementation and Analysis of Results.

### 2.3.1. Investigation

As a starting point for the project, we had to carry out a previous study of different user classification techniques through different research articles on the subject. In one of them [2], we found a comparison between the results of several reserach papers and decided to focus on a subset of them.

After studying and analyzing this subset of techniques, we decided to stick with a few, which we have implemented and analyzed during the work:

- Manhattan detector [6]: The distances between each characteristic of a new entry and the characteristics of the average vector of a subject must all be less than a concrete threshold.
- Mean and standard deviation [7]: The sum of the differences between a new entry and the average vector of a user should be less than the mean of the differences, calculated between the mean vector and previous entries; plus *n* standard deviations from those differences.

- Z-scores [8]: a new entry must have a minimum of values within confidence intervals of the subject who claims to belong.

### 2.3.2. Data Collection

A complete keystroke dynamics system requires a data capture application and a large number of key inputs generated by multiple people. Given the complexity of creating our own from scratch, we decided to focus on the analysis of the different techniques using existing data sets and leave for the future the creation of our own system.

The process of finding a data set was very hard but, after much searching, we found a large dataset, which had more than twenty thousand entries from 51 different users [14]. Each of these inputs records the existing dwell and flight times by entering a specific password, in addition to the user who performed it, its session number and repetition number.

This data set was encoded using three different formats: Excel, Plain Text and CSV. Because of this, we decided to do some research on what kind of format will allow us to extract the data more easily for our project. After several tests, we concluded that the most appropriate format was CSV. Then, we implemented a program for extracting data, which sorts and classifies the different biometric characteristics of each input and stored in a data structure.

However, before beginning to code any of the chosen techniques, we had to create a program that divided our data set into two parts: training and testing. This is because the classifiers analyze the data of the training set so that they can then check if a user in the test set is who they say they are.

### 2.3.3. Implementation

Before implementing each technique, we conducted an in-depth study of each of them to avoid possible future errors. Once this is done, we begin the implementation of the algorithms in the following order:

1. Detector Manhattan
2. Mean and standard deviation
3. Z-scores

To refer to them throughout all the work, we will use the following notations:

- Detector Manhattan ≡ MAN authentication system.
- Mean and standard deviation ≡ MEANSTDV authentication system.
- Z-scores ≡ ZSCORE authentication system.

All of them follow the same pattern of operation:

1. Initialization of the algorithm with its specific parameters (different according to the technique).
2. Training.

3. Test.
4. Return of the result obtained.

### 2.3.4. Analysis of Results

After completing the implementation of the three classifiers, we performed a battery of tests with the dataset and tested different combinations of characteristics to consider and parameters. The results of each combination are stored in a file for later exhaustive analysis and concrete conclusions. This analysis can be seen in Section 5.

# 3. Autenticación mediante dinámica del tecleo

La dinámica del tecleo ofrece una gran cantidad de posibilidades y técnicas a la hora de evaluar a un usuario. Por tanto, en este apartado explicaremos todo lo necesario para comprender el funcionamiento básico de un sistema de dinámica de pulsación y los datos que utiliza.

Este tipo de sistema sigue la siguiente estructura:

1. **Extracción de características:** obtenemos los datos más relevantes de cada usuario para poder utilizarlos más tarde.
2. **Clasificación de usuarios:** aplicamos diferentes cálculos a los datos obtenidos para decidir si un usuario es auténtico o es un impostor.
3. **Evaluación del rendimiento:** realizamos una clasificación de usuarios masiva para obtener métricas sobre la eficiencia del sistema.

A continuación, explicaremos de manera detallada cada una de las partes de la estructura anterior.

## 3.1. Características Biométricas

Para poder clasificar a un usuario, primero necesitamos extraer determinadas características de él, para que más tarde nos permitirán comprobar si realmente es quien dice ser. En el ámbito de la dinámica del tecleo, existen distintas características extraíbles de un usuario, sin embargo, las más utilizadas e importantes son las que vamos a explicar a continuación [1,3].

***Di-graphs***
Tiempo que pasa desde que comenzamos a pulsar una tecla hasta que pulsamos la siguiente [1,3]. Existen dos tipos bien diferentes:

1. ***Dwell time* (DT) o Tiempo de Presión:** Es el tiempo que tenemos pulsada una tecla, es decir, desde que pulsamos una tecla hasta que la soltamos [3].

2. ***Flight time* (FT) o Tiempo de Vuelo:** Es el tiempo que transcurre en el paso de una tecla a otra [3]. Existen cuatro tipos distintos según cómo lo calculamos:
    - $F_{type1}$ *(up-down):* desde que soltamos una tecla hasta que pulsamos la siguiente. También denominado FT1.
    - $F_{type2}$ *(up-up):* desde que soltamos una tecla hasta que soltamos la siguiente. También denominado FT2.
    - $F_{type3}$ *(down-down):* desde que pulsamos una tecla hasta que pulsamos la siguiente. También denominado FT3.

- $F_{type4}$ *(down-up):* desde que pulsamos una tecla hasta que soltamos la siguiente. También denominado FT4.

*F*

En la siguiente imagen, podemos ver cómo se calculan los dos tipos de *di-grahps*, siendo "J" e "Y" dos letras consecutivas:

- El *dwell time* de la tecla "J" se calcula restando el momento en el que la soltamos (R) menos el momento en que la pulsamos (P).
- $F_{type1}$ se calcula restando el momento en que presionamos "Y" menos el momento en que soltamos "J".
- $F_{type2}$ se calcula restando el momento en que soltamos "Y" menos el momento en que soltamos "J".
- $F_{type3}$ se calcula restando el momento en que presionamos "Y" menos el momento en que presionamos "J".
- $F_{type4}$ se calcula restando el momento en que soltamos "Y" menos el momento en que presionamos "J".

Debido a la necesidad de presionar varias teclas a la vez para introducir caracteres específicos, tanto $F_{type1}$ como $F_{type2}$ pueden ser negativos. <u>*Ejemplo*:</u> si queremos "J" sea mayúscula, necesitamos mantener la tecla "Shift" al menos hasta el momento que pulsemos "J", provocando esto que $F_{type1}$ y $F_{type2}$ de "Shift" a "J" sean menores o iguales que cero.

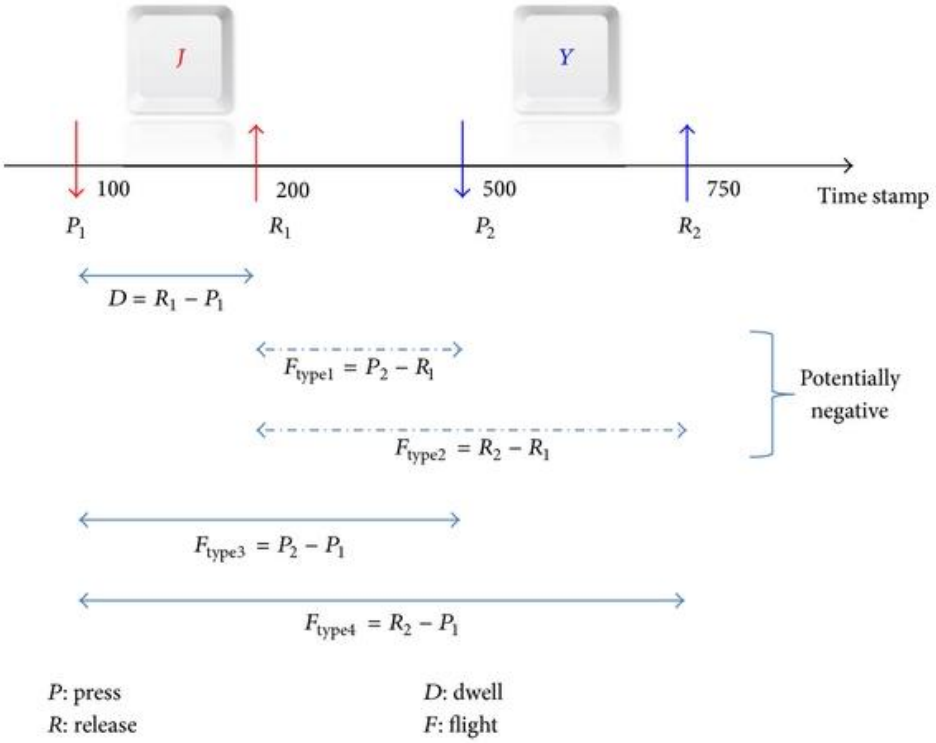



Imagen extraída del artículo [3].

***N-graphs* (ET)**

Es el tiempo transcurrido entre tres o más pulsaciones consecutivas. "N" determina el número de pulsaciones consecutivas. La fórmula más utilizada para calcular *N-graphs* es la siguiente [3]:

$$\mathrm{ET}_k = P_{k+n} - P_k,$$

donde “P” indica una pulsación, “n” es el número de pulsaciones consecutivas y “k” representa la posición de la pulsación en la que se calcula el *N-graph*.

Como es obvio, por cada característica que extraigamos de cada usuario, tendremos varios valores. Por tanto, cada entrada al sistema será transformada en un conjunto de vectores o listas, tan grande como características queramos utilizar.

Durante este trabajo, hemos estado trabajando con todos los tipos de características anteriores. Sin embargo, decidimos realizar el análisis utilizando todas ellas excepto los *N-graphs*, debido al elevado número de combinaciones posibles.

## 3.2. Métodos de Clasificación

Existen múltiples métodos de clasificación de usuarios, sin embargo, la mayoría suele seguir un mismo patrón de actuación para poder evaluar si un usuario es genuino:

1. *Entrenamiento:* genera una plantilla de características para cada usuario mediante los datos obtenidos anteriormente. Dicha plantilla será el mecanismo que usaremos para comprobar si una nueva entrada corresponde a ese usuario o no.
2. *Test:* comparamos las plantillas obtenidas con nuevas entradas. Dependiendo del grado de similitud entre una nueva entrada y una plantilla, esa entrada será aceptada en el sistema.

A pesar de que la mayoría siga este patrón, cada uno utiliza diferentes técnicas para obtener la plantilla y determinar el grado de similitud. Aun así, podemos distinguir varios grupos según el tipo de técnicas que utilizan: estadísticas, distancia, clustering, aprendizaje automático, redes neuronales, programación evolutiva, etc.

En este Trabajo de Fin de Grado nos hemos centrado en el estudio de técnicas pertenecientes a dos de los grupos más importantes:

- Estadísticas:
  - Sistema de autenticación MAN.
  - Sistema de autenticación MEANSTDV.
  - Sistema de autenticación ZSCORE
- Aprendizaje automático (*Machine learning*):
  - K-means, Euclidean.
  - K-Nearest Neighbor.
  - Bayesian y Mínimo.

Aclarado qué tipos de técnica hemos estudiado, en el siguiente apartado definiremos un conjunto de expresiones que más tarde utilizaremos en la explicación de varias técnicas.

### 3.2.1. Notación

Cada técnica que hemos estudiado utiliza una notación diferente, y para este trabajo hemos tenido que unificarlas. En este apartado resumimos la notación que usaremos para la los cálculos de las distintas técnicas.

- $i$ : usuario concreto.
- $e$ : vector de valores de una entrada.
- $l$ : característica actual (DT, FT1, FT2, FT3, FT4).
- $n$ : número de usuarios.
- $m$ : número de entradas de entrenamiento.
- $e_{ij}$ : vector de valores de la entrada $j$ del usuario $i$.
- $e_{ijl}$ : vector de valores correspondiente a la característica $l$ en una entrada $j$ de un usuario $i$.
- $e_l$ : vector de valores correspondiente a la característica $l$ del vector $e$.
- $e_{jl}$ : vector de valores correspondiente a la característica $l$ en una entrada $j$.
- $\mu$ : vector de valores medios.
- $\mu_i$: vector de valores medio del usuario $i$.
- $\mu_{il}$ : vector medio correspondiente a la característica $l$ del usuario $i$.
- $\sigma$ : vector de desviaciones.
- $\sigma_i$ : vector de desviaciones del usuario $i$.
- $\sigma_{il}$ : vector de desviaciones que corresponde la característica $l$ del usuario $i$.
- $\mathrm{dist}_{lk}(e,i)$: distancia entre la entrada $e$ y el usuario i en la posición $k$ de la característica $l$.
- $T_e$ : tamaño del vector de valores de una entrada $e$.
- $T_l$ : tamaño del vector de valores de una característica $l$.

Dado un vector X se utilizará la notación X(k) para indicar el elemento k-ésimo.

Aclarada la noción que seguirán las distintas técnicas, pasamos a su explicación.

### 3.2.2. Técnicas Estadísticas

Los métodos estadísticos son los más comunes debido a su baja complejidad, fácil implementación y poca sobrecarga [3]. Estas técnicas se centran en medidas estándares como la media, la mediana y la desviación típica. A continuación, explicaremos varias técnicas de este tipo:

#### Sistema de autenticación MAN [6]

Esta técnica se basa en el cálculo de las distancias entre las características de una nueva entrada con las características de una entrada media previamente generada.

Lo primero que debemos hacer es entrenar el clasificador con los datos obtenidos anteriormente para generar una plantilla por cada usuario del sistema. La plantilla de un usuario almacenará la media y la desviación de cada valor de cada una de sus característica, es decir, un conjunto de vectores con todas las medias y otro conjunto de vectores con todas las desviaciones.

La media y la desviación de cada valor de una característica cualquiera se calculan de la siguiente forma:

$$\mu_{il}(k) = \frac{1}{m}\sum_{j=1}^{m} e_{ijl}(k)$$

$$\sigma_{il}(k) = \frac{1}{m-1}\sum_{j=1}^{m} \left| e_{ijl}(k) - \mu_{il}(k) \right|$$

Finalizado el entrenamiento, podemos realizar la evaluación de una nueva entrada. Para comprobar si una entrada corresponde a un usuario concreto, se obtienen las características utilizadas de la entrada y se calcula la distancia entre ellas y la plantilla de ese usuario en unidades de desviación. Dicha distancia se calcula de la siguiente forma:

$$D_l(e,\, i) = \frac{1}{T_l}\sum_{k=1}^{T_l} dist_{lk}(e,\, i)$$

donde *D* es la distancia entre la entrada *e* y el usuario *i* en la característica *l*, que a su vez se calcula utilizando la siguiente fórmula:

$$dist_{lk}(e,\, i) = \frac{\left| e_l(k) - \mu_{il}(k) \right|}{\sigma_{il}(k)}$$

Como se puede observar, $dist_{lk}(e, i)$ mide la distancia entre la entrada *e* y el usuario i en la posición *k* de la característica *l*.

Una vez obtenidas las distancias de cada característica, debemos comprobar como de próximas están de la plantilla. Para ello, cada característica deberá tener establecido un umbral. Estos umbrales determinan lo cerca que debe estar una entrada para ser aceptada. En esta técnica, si todas las distancias son menores o iguales que sus respectivos umbrales, la entrada es aceptada en el sistema.

*EJEMPLO 1*: Para un usuario *i* cuya contraseña es "**bft7**", tenemos 3 entradas de entrenamiento {*j1, j2, j3*} donde solo se ha obtenido la característica *dwell time*. Los valores para cada entrada son los siguientes:

*j1* = {0.0950, 0.0797, 0.0636, 0.0803}
*j2* = {0.1256, 0.0808, 0.0914, 0.0903}
*j3* = {0.0934, 0.0940, 0.0697, 0.0985}

Con estos datos, calculamos la media y la desviación para cada *dwell time* de la contraseña. Como demostración, calcularemos el de la letra "**f**":

$$\mu = \frac{0.0797 + 0.0808 + 0.0940}{3} = 0.0848$$

$$\sigma = \frac{|0.0797 - 0.0848| + |0.0808 - 0.0848| + |0.0940 - 0.0848|}{3 - 1} = 0.0091$$

Como cabe esperar, calculamos estos cálculos para todas las letras, dándonos un vector de medias y otro de desviaciones:

$\mu$ = {0.1046, 0.0848, 0.0749, 0.0906}

$\sigma$ = {0.0209, 0.0091, 0.0165, 0.0092}

Con estos datos, ya podemos clasificar una nueva entrada. En este caso, vamos a valorar la entrada *e* = {0.0929, 0.0818, 0.0747, 0.0776}. Para ello, necesitamos comparar cada valor de *e* con la media y desviación del usuario *i*. En este caso vamos a mostrar los cálculos para la letra "**b**":

$$dist_{DT\,0}(e, i) = \frac{|0.0929 - 0.1046|}{0.0209} = 0.0559$$

Los cálculos para el resto de valores serían muy parecidos, por lo que pasamos al cálculo de la distancia final:

$$D_{DT}(e, i) = \frac{0.0559 + 0.329 + 0.012 + 1.413}{4} = 0.5782$$

Esta es la distancia existente entre la entrada *e* y el usuario *i* en la característica *dwell time*. Ahora tendríamos que compararla con el umbral de esta característica. Si este resultado fuese menor o igual que el umbral, la entrada sería aceptada.

En este caso solo hemos utilizado la característica *dwell time*, pero si utilizáramos alguna más, tendríamos que repetir este proceso para cada una de ellas.

Debido a la simplicidad de los cálculos de esta técnica y su fácil implementación, decidimos utilizarla en nuestro estudio.

## Sistema de autenticación MEANSTDV [7]

Esta técnica calcula la diferencia entre una nueva entrada y un vector medio $\mu$ para después compararla con la media y desviación típica de las diferencias entre el vector medio $\mu$ y las entradas con las que fue generado.

Como ocurre generalmente, la técnica comienza generando una plantilla por cada usuario. El entrenamiento de la plantilla sigue varios pasos:

1. Primero, generamos el vector medio $\mu$ de cada usuario. Este almacena, por cada característica extraída, la media de cada uno de sus valores obtenidos de las entradas de un usuario almacenadas en el sistema.
2. Una vez tenemos el vector medio de todos los usuarios, volvemos a recorrer todas las entradas de cada usuario y calculamos cuánto de diferentes son de sus correspondientes vectores medios. La magnitud de diferencia entre un vector medio $\mu$ y una entrada se calcula según la siguiente norma:

$$||\mu - e||$$

dada por:

$$\sum_{k=1}^{T_e} |\mu(k) - e(k)|$$

Esto quiere decir que se realiza la suma del valor absoluto de la diferencia de cada valor existente de todas las características extraídas.

3. Cuando ya hemos obtenido todas las diferencias entre un vector medio y sus entradas correspondientes, calculamos la media y la desviación típica de esas diferencias.

De esta forma, la plantilla de cada usuario del sistema estará formada por su vector medio $\mu$ junto con la diferencia media y la desviación típica de las diferencias de distancias entre su vector medio y sus entradas anteriores.

Una vez obtenidas todas las plantillas, damos el entrenamiento por terminado y pasamos a clasificar nuevas entradas. Para ello, tenemos que calcular la diferencia entre la nueva entrada con el vector medio del usuario a quien dice corresponder, mediante la norma que hemos visto antes. Dicha diferencia deberá ser estrictamente menor que un umbral.

El umbral es único de cada usuario y se calcula sumándole a la media de su plantilla un número concreto de desviaciones típicas. Ese número de desviaciones típicas es común para todos los usuarios del sistema y será establecido al inicializar la técnica.

*EJEMPLO 2:* siguiendo con los datos del ejemplo 1, vamos a aplicarles esta técnica. El vector medio $\mu$ se calcula de la misma forma que en el ejemplo 1, por tanto sus valores son los mismos:

$\mu$ = {0.1046, 0.0848, 0.0749, 0.0906}

Teniendo ya el vector medio, lo siguiente es calcular las diferencias entre ese vector medio y las entradas de entrenamiento. El cálculo de la distancia entre $\mu$ y *j1* es de la siguiente forma:

$$||\mu - j1|| = |0.095 - 0.1046| + |0.0797 - 0.0848| + \\ + |0.0636 - 0.0749| + |0.0803 - 0.0906| = 0.0363$$

Los cálculos de las diferencias para las otras entradas son iguales, quedando los siguientes resultados:

$$||\mu - j1|| = 0.0363$$
$$||\mu - j2|| = 0.0415$$
$$||\mu - j3|| = 0.0335$$

El siguiente paso es calcular la media y desviación típica de estas distancias:

$$\mu\ distancias\ = \frac{0.0363 + 0.0415 + 0.0335}{3} = 0.0371$$

$$\sigma\ distancias\ = \sqrt{\frac{(0.0363 - 0.0371)^2 + (0.0415 - 0.0371)^2 + (0.0335 - 0.0371)^2}{3 - 1}} = 0.004$$

Con estos datos, ya podemos clasificar una nueva entrada. En este caso utilizaremos la entrada *e* del ejemplo 1. Para clasificar e, necesitamos calcular la diferencia respecto al vector medio $\mu$ del usuario *i*:

$$||\mu - e|| = 0.279$$

Para decidir si e es aceptada deberá cumplir la siguiente condición:

$$0.279 < 0.0371 * numDesv$$

donde *numDesv* tiene 0 como valor más estricto y según aumenta, el sistema es más permisivo.

Debido a los similares cálculos con la técnica anterior y sus prometedores resultados, decidimos implementar esta técnica y utilizarla en nuestro estudio.

## Sistema de autenticación ZSCORE [8]

Esta técnica acepta una nueva entrada dependiendo del número de outliers que tenga respecto de los intervalos de confianza del usuario a quien dice pertenecer.

Como en las otras técnicas, el entrenamiento de esta técnica consiste en generar una plantilla de datos de cada usuario del sistema. Primero, calculamos la media y la desviación típica de todos los valores de cada característica extraída por el sistema. Con ellas, calcularemos una serie de intervalos de confianza en los cuales debería estar cada valor de una nueva entrada. Cada intervalo se calcula de la siguiente manera:

$$\mu_i(k) \pm z * \sigma_i(k)$$

donde $\mu$ es la media, $\sigma$ la desviación típica, *z* la distribución estándar y *k* la posición del valor actual. El valor de la distribución estándar se establecerá previamente al inicializar la técnica.

Una vez obtenidos todos los intervalos de confianza de todos los usuarios, el entrenamiento concluye y pasamos a clasificar nuevas entradas.

Para clasificar una entrada, compararemos si los valores de sus características están dentro de sus intervalos de confianza. Cada valor dentro de su intervalo se considerará como un acierto. Para que una entrada sea considerada como válida, deberá tener un mínimo de aciertos (umbral).

*EJEMPLO 3:* partiendo de los datos del ejemplo 1, vamos a realizar una demostración de esta técnica. Lo primero que tenemos que hacer es calcular el vector medio y el vector de desviaciones típicas de las entradas de entrenamiento. El vector medio $\mu$ tendrá los mismos valores que en el ejemplo 1, sin embargo el vector de desviaciones $\sigma$ no. Esto se debe a que la primera técnica no utiliza la fórmula de la desviación típica. El vector medio y el vector de desviaciones típicas $\sigma$ resultantes son:

$\mu$ = {0.1046, 0.0848, 0.0749, 0.0906}
$\sigma$ = {0.018, 0.0079, 0.0146, 0.0091}

Con estos datos, junto con la distribución estándar, calculamos el intervalo de confianza cada valor de la entrada. En este caso, vamos a establecer el valor de la distribución estándar en 3. Los intervalos resultantes son:

$int_1$= [0.0506, 0.1586]
$int_2$= [0.0611, 0.1085]
$int_3$= [0.0311, 0.1187]
$int_4$= [0.0633, 0.1179]

Una vez tenemos los intervalos de confianza, podemos valorar una nueva entrada. En este caso vamos a utilizar la entrada e del ejemplo 1. Para ver si es aceptada o no, comprobamos cuántos de sus valores están dentro de sus respectivos intervalos de confianza. Dependiendo de cuantos estuvieran dentro, la entrada sería aceptada o rechazada. En este caso, todos los valores están dentro de sus intervalos, así que la entrada sería aceptada aunque el número de aciertos mínimo fuese 4 (umbral más estricto).

Debido a la fácil implementación de esta técnica y a sus prometedores resultados, decidimos utilizarla en nuestro estudio. En él, decidimos establecer el valor de la distribución estándar en 2, debido a que abarca más del 95% de los valores y a sus prometedores resultados en una batería de pruebas previa. Además, establecimos como umbral el porcentaje mínimo de aciertos que debe tener una entrada para ser aceptada.

### 3.2.3. Técnicas de Aprendizaje Automático

El aprendizaje automático es un proceso de inducción del conocimiento que consiste en crear programas capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos [15]. En muchas ocasiones el campo de actuación del aprendizaje automático se solapa con el de la estadística computacional, ya que las dos disciplinas se basan en el análisis

de datos. Sin embargo, el aprendizaje automático también se centra en el estudio de la complejidad computacional de los problemas. Esta categoría incluye técnicas como el uso de redes neuronales, árboles de decisión e informática evolutiva.

A continuación, explicaremos varias técnicas de este tipo:

## K-Means, Euclidean [9]

Esta técnica se basa en el algoritmo K-means, el cual tiene como objetivo la partición de un conjunto de datos (entradas en este caso) en diferentes grupos o clases [10]. Además, aplican la fórmula de la distancia Euclídea para el cálculo de todas las distancias.

Primero, realizamos el proceso de entrenamiento del algoritmo, el cual determina a qué grupo pertenece cada una de las entradas almacenadas en el sistema y establece una entrada promedio por cada grupo existente. El número de grupos se establece al inicializar el algoritmo.

Después clasificaremos una nueva entrada, comprobando cuál es su grupo más cercano. Para ello, calcularemos la distancia de la entrada a las distintas entradas promedio y nos quedaremos con el grupo al que dicha distancia sea menor.

Finalmente, compararemos la distancia al grupo más cercano con un umbral. Este umbral será exclusivo de cada grupo y se calcula de la siguiente forma:

$$U \times \frac{1}{N_g} \sum_{j \in E_g} dist(j, Prom(g))$$

donde *U* es un coeficiente preestablecido y se multiplica por la distancia media entre:

- Todos las entradas *j* pertenecientes al grupo *g* más cercano.
- Y la entrada promedio de ese grupo (*Prom*(*g*)).

Siendo $E_g$ el conjunto de entradas del grupo *g* y $N_g$ el número de entradas de dicho grupo.

Esto quiere decir que si la distancia de la nueva entrada al prototipo de entrada del grupo más cercano es estrictamente menor que su umbral, la entrada será considerada como válida.

Por motivos de falta de tiempo, no hemos podido incluirla en nuestro estudio.

## K-Nearest Neighbors [11]

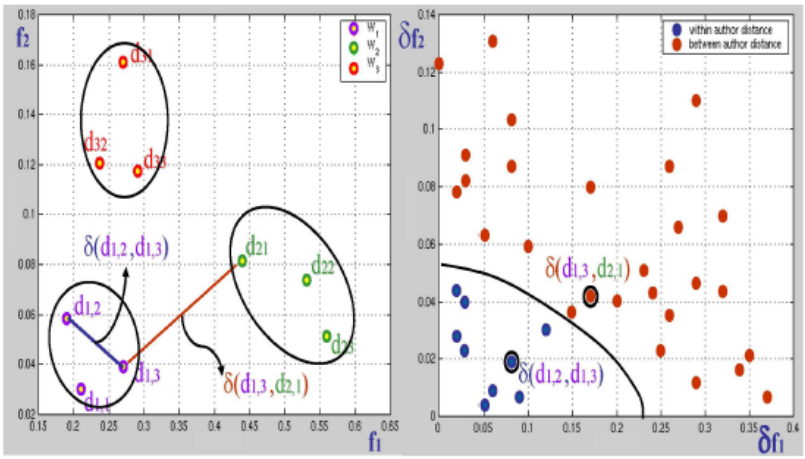
Esta técnica se basa en el algoritmo de mismo nombre, el cual, clasifica que un nuevo dato pertenece a la clase que contiene el mayor número de vecinos dentro de un rango *k* [12].

Sabiendo esto, esta técnica utiliza únicamente dos clases:

- Intrapersonal (X⊕): el usuario es genuino.
- Interpersonal (X⊖): el usuario es un impostor.

Sin embargo, esta técnica no clasifica una entrada en una de esas dos clases, sino que calcula todas las distancias entre cada par de entradas y las clasifica en esas clases mediante el algoritmo. En la siguiente imagen, podemos observar cómo pasamos de una representación con varias clases, utilizando entradas, a una representación con solo dos clases utilizando distancias entre entradas:

- La representación de la izquierda corresponde a un espacio para tres usuarios, donde las entradas de cada uno están coloreadas de diferentes colores, uno por cada usuario.
- La representación de la derecha corresponde a un único usuario, donde las distancias entre sus entradas están coloreadas de azul, mientras que las distancias de sus entradas a las entradas del resto de usuarios están en rojo.




(a) Feature space (b) Feature-difference space

Imagen perteneciente al artículo [11].

Siendo $d_{ij}$ el vector de características de la j-ésima entrada del i-ésimo usuario, el conjunto de distancias intrapersonales se calculan de la siguiente manera:

$$X\oplus = \left| d_{ij} - d_{ik} \right|$$ donde i va de 1 a *n* y *j*,*k* de 1 a *m*, para j distinto de *k*.

Mientras que el conjunto de distancias interpersonales se calcula así:

$$X\ominus = \left| d_{ij} - d_{kl} \right|$$ donde *i*,*k* van de 1 a *n*, para i distinto de k, y *j*,*l* de 1 a *m*.

Donde *n* es el número de personas y *m* es el número de muestras por persona.

La clasificación de las entradas obtenidas en esas dos clases, por cada usuario existente, corresponde al entrenamiento.

Para clasificar una nueva entrada, primero calculamos la distancia entre esta y otra entrada anterior del usuario que dice ser. Después, le pasamos esta distancia al algoritmo, el cual, determinará si esta es intrapersonal o interpersonal, es decir, si la entrada es válida o no.

No pudimos utilizarla en nuestro trabajo por falta de tiempo.

**Bayesian y Mínimo [13]**

Se basa en el algoritmo de Bayes, algoritmo basado en el Teorema del mismo nombre. En un sistema de clases donde cada dato pertenece a una clase concreta, este algoritmo determina a qué clase pertenece un nuevo dato.

En esta técnica cada clase corresponderá con cada usuario del sistema. De esta forma, cada usuario $i$ se compondrá de un vector de características medio $\mu$ y una matriz de covarianza $Cov_i$. Ambas cosas se calculan de la siguiente forma:

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} d_{ij}$$

$$Cov_i = \frac{1}{m}\sum_{j=1}^{m} d_{ij} d_{ij}^{\ t} - \mu_{ij}\mu_{ij}^{\ t}$$

Una vez calculadas la media y la matriz de covarianza para todos los usuarios, podemos clasificar una nueva entrada. Para ello, se calcula la probabilidad gaussiana de la nueva entrada para cada usuario del sistema. Su fórmula es la siguiente:

$$P_i(e) = (2\pi)^{-T_e}\, exp[(-1/2)(d_e - \mu_i)^t Cov_i^{-1}(d_e - \mu_i)]$$

Una vez calculadas todas, nos quedaremos con el usuario cuya probabilidad sea la mayor. La entrada únicamente será aceptada si el usuario al que corresponde esa probabilidad es el mismo que afirmaba ser la entrada *e*.

No pudimos aplicarla en nuestro estudio por falta de tiempo.

## 3.3. Evaluación del Rendimiento

Como hemos visto, el proceso de determinación de los umbrales es muy importante para el correcto funcionamiento del sistema. Dependiendo de qué umbrales establezcamos, aceptaremos a usuarios falsos y rechazaremos a usuarios genuinos. Por ejemplo, si establecemos umbrales poco exigentes, aceptaremos a muchos usuarios falsos; y si utilizamos umbrales demasiado estrictos, rechazaremos a usuarios auténticos.

A raíz de este concepto, surgen dos tasas de error que evalúan el rendimiento y la eficacia de un sistema de dinámica de pulsación [1,3,4,5]:

- **Tasa de Falso Rechazo (FRR):** es la probabilidad de que se determine como impostor a un usuario genuino. Se calcula dividiendo el número de usuarios falsamente rechazados entre el número total de usuarios auténticos que acceden al sistema.
- **Tasa de Falsa Aceptación (FAR):** es la probabilidad de que se determine como genuino a un impostor. Se calcula dividiendo el número de impostores aceptados en el sistema entre el total de impostores que intentan acceder a él.

Un sistema será más seguro cuanto menores sean ambas tasas de error.

Como podemos suponer, para diferentes valores de los umbrales, obtendremos diferentes valores de ambas tasas. De este modo, si calculamos múltiples valores de FRR y FAR para diferentes umbrales y los representamos gráficamente, obtendremos dos curvas, cada una de ellas correspondiente a una tasa de error [1,4]. Esta representación sirve para obtener una nueva tasa: el punto donde se cortan las dos curvas se denomina Tasa de Error Igual (ERR) o Tasa de Error de Cruce (CER) [1,3,4,5]. La ERR es una tasa de rendimiento independiente del umbral [4]. Cuanto menor sea la ERR, mejor será el rendimiento del sistema [1,4].

En la siguiente imagen podemos ver un ejemplo gráfico de los tres tipos de tasas anteriores:

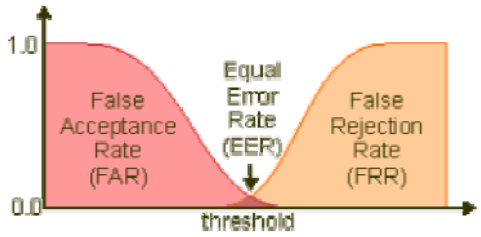



Imagen obtenida del artículo [4].

En la imagen, vemos como las tres tasas fluctúan para diferentes valores del umbral (*threshold*). Los valores de cada tasa están entre 0 y 1 debido a todas ellas son probabilidades.

En el caso de que quisiéramos calcular la ERR a partir de una tabla con distintos valores de FRR y FAR para diferentes umbrales, puede ocurrir que no dispongamos del caso en que FRR = FAR. Por tanto, para calcular el ERR tendríamos que seguir el siguiente proceso:

- Obtener las FRR y FAR más cercanas.
- Una vez obtenidas, ERR se calcularía así:

$$ERR = \frac{FRR + FAR}{2}$$

A pesar de la existencia de otras métricas como son la tasa de falsas alarmas de fallo cero (ZMFAR) o la tasa de falsa aceptación cero (la FRR cuando FAR es igual a cero) [1], las tres anteriores son las más conocidas y utilizadas en este campo de la biometría. Y por tanto, son las que hemos utilizado en nuestro estudio.

# 4. Propuesta Software

En este apartado comentaremos los distintos detalles del programa implementado durante el proyecto.

Para su implementación, hemos utilizado Java (versión 8, actualización 112) como lenguaje de programación y Eclipse (versión 4.4.1) como entorno de desarrollo. Además, decidimos utilizar Maven como gestor de proyecto para facilitar el uso de librerías externas cuando las necesitáramos.

En cuanto a la estructura, creamos paquetes y clases según los necesitábamos. De esta forma, cada paquete corresponde a una funcionalidad concreta del proyecto. El programa contiene los siguientes paquetes: `main`, `pruebas_iniciales`, `extraccion`, `muestreo`, `analisis` y `evaluacion`.

Todo el código implementado junto con todos los resultados obtenidos, están disponibles en el siguiente repositorio de GitHub[1].

A continuación, explicaremos el proceso de implementación del programa junto con algunos detalles importantes.

## 4.1. Lectura y tratamiento del conjunto de datos

Para comenzar la implementación del sistema, necesitábamos extraer los datos del conjunto de entradas que habíamos conseguido. Sin embargo, primero era necesario hacer un pequeño estudio sobre el formato del fichero que íbamos a utilizar, ya que disponíamos del conjunto de datos en tres formatos: CSV, texto plano y Excel. Para ello, investigamos e implementamos diferentes formas de extraer los datos de los ficheros. Finalmente, concluimos que el formato CSV era el más sencillo y útil ya que nos permitía extraer todos los valores de una entrada en una cadena de caracteres, separados por comas. El paquete `pruebas_iniciales` contiene clases con algunos métodos de extracción que investigamos.

Una vez teníamos claro que formato de fichero utilizar, comenzamos con la extracción de los datos del conjunto de entradas. Dicho conjunto está formado por más de veinte mil entradas, cada una de ellas formada por:

- Usuario de la entrada
- Número de sesión
- Número de repetición
- Múltiples valores de DT, FT1 y FT3

Este conjunto de entradas había sido formado en el estudio [14]. En él, 51 usuarios habían realizado diferentes entradas de una misma contraseña. Cada usuario había realizado 8 sesiones, repitiendo la contraseña 50 veces por sesión.

[1] https://github.com/ferrivi14/TFG-Keystroke_Dynamics

Conocida la estructura del conjunto y el tipo de valores que contenía, decidimos extraer todos y cada uno de ellos y, además, calcular algunos nuevos. Como el conjunto ya disponía de los tiempos de presión y dos tipos de tiempos de vuelo, creamos métodos que calculan los otros dos tipos de tiempos de vuelo y los *tri-graph* a través de los datos que ya teníamos. Sin embargo, necesitábamos almacenarlos en una estructura concreta que nos facilitase su uso a lo largo del programa. Por tanto, creamos la clase `Password`, la cual almacena el usuario, la sesión, la repetición y las seis características biométricas extraídas de una única entrada. Esta clase ha sido nuestro pilar central en el desarrollo del sistema y está almacenada en el paquete `main`.

Sabiendo la forma de extraer y almacenar todos los datos de una entrada, nos fue sencillo implementar una función que lo hiciese para todo un conjunto de entradas. Dicha función recibe la ruta de un fichero CSV y devuelve una lista de `Password`. Todo este proceso se desarrolla en la clase `Extraccion` del paquete `extraccion`.

Antes de ponernos con la implementación de las técnicas de clasificación, tuvimos que dividir el conjunto de datos en dos partes:

- Entrenamiento: conjunto de entradas con las que entrenaremos cada clasificador.
- Test: conjunto de entradas que clasificaremos para obtener el rendimiento de cada técnica.

Para ello, creamos un método que dados la ruta de un fichero CSV y un porcentaje, genera un fichero de entrenamiento y otro de test según el porcentaje establecido. Este porcentaje determina el número de entradas de cada usuario que contendrá el conjunto de entrenamiento, correspondiendo el resto de las entradas al conjunto de test. Este es el método principal de la clase `Muestreo` del paquete `muestreo`.

## 4.2. Implementación de las técnicas de clasificación

Para comenzar, decidimos crear una interfaz `Clasificador` que más tarde implementarían las clases correspondientes a cada técnica. Esto se debe a que las tres técnicas implementadas siguen una misma estructura: entrenamiento y test. De tal modo que la interfaz `Clasificador` únicamente tiene dos métodos:

- `entrenar`(): entrena el clasificador correspondiente mediante una lista de `Password` que recibe.
- `testear`(): devuelve los resultados obtenidos de clasificar una lista de `Password` dada.

En este momento, antes de comenzar a implementar ninguna técnica, establecimos el formato de los resultados. Estos serian guardados en una nueva clase `Resultados`, en la cual almacenamos los valores del FRR, FAR y ERR de manera sencilla.

Establecido el formato de los resultados, comenzamos la implementación de tres clases nuevas, una por cada técnica: `Manhattan_detector`, `Mean_Standar_deviation` y `Z_scores`.

**Clase Manhattan_detector**

Es la clase que implementa el sistema de autenticación MAN. Fue la más costosa de implementar debido a que no teníamos una estructura clara que seguir por debajo del proceso de entrenamiento y test. Pero una vez terminada, nos ayudó enormemente a la hora de implementar las otras dos técnicas. A parte de los métodos de entrenar y testear, que comparten todas, podemos destacar otros importantes:

- `constructor`(): establece que características biométricas se van a utilizar y los umbrales pertenecientes a cada una de ellas.
- `media`(): calcula los vectores medios de los usuarios existentes en un conjunto de `Password` que recibe.
- `desviacionTipica`(): calcula los vectores de desviaciones de los usuarios existentes en un conjunto de `Password` que recibe.
- `verificarPassword`(): determina si una `Password` pertenece a un sujeto concreto.

**Clase Mean_Standar_deviation**

Clase que implementa el sistema de autenticación MEANSTDV. Aparte de los métodos propios de la interfaz `Clasificador`, podemos destacar los siguientes:

- `constructor`(): establece las características biométricas extraídas de cada `Password` y el número de desviaciones típicas necesario para calcular los umbrales.
- `calculoMs`(): calcula el vector medio para todos los usuarios incluidos en una lista de `Password`.
- `diferenciasEntrenamiento`(): calcula las diferencias entre una lista de Password y sus correspondientes vectores medios.
- `calculoMediasYDesviaciones`(): calcula para cada usuario del entrenamiento, la media y desviación típica de sus diferencias entre su vector medio y sus `Password` correspondientes.
- `diferenciaEntrePasswords`(): calcula la diferencia entre dos `Password`.
- `calcularUmbrales`(): calcula el umbral para cada usuario del entrenamiento.
- `verificarPassword`(): clasifica si una `Password` pertenece a un determinado sujeto.

**Clase Z_scores**

Clase que implementa la técnica "Z-scores". Debido a las necesidades de esta técnica, decidimos implementar tres clases complementarias:

- Clase `Intervalo`: nos permite establecer el rango de un intervalo y comprobar si un valor está dentro de él.
- Clase `IntervalosPassword`: almacena las listas de intervalos de un sujeto imitando la estructura de `Password`.

- Clase `Aciertos`: almacena el número de aciertos y permite determinar si son suficientes respecto a un porcentaje concreto.

A parte de las funciones comunes con las clases de las otras técnicas, los métodos más importantes son:

- `constructor`(): determina las características extraídas que se usarán a lo largo de la técnica y los valores de la distribución estándar y del porcentaje de aciertos mínimos.
- `media`(): calcula el vector medio de cada usuario existente en un conjunto de `Password` dado.
- `desviacionTipica`(): calcula el vector de desviaciones típicas de cada usuario existente en un conjunto de `Password` dado.
- `calculoIntervalosPassword`(): calcula los intervalos de un sujeto concreto.
- `calculoAciertosCaracteristica`(): dado un vector de valores de una característica concreta y su lista de intervalos, determina el número aciertos.

Todo las clases referentes a la implementación de las distintas técnicas están almacenadas en el paquete `analisis`.

## 4.3. Generación de resultados

Una vez concluida la implementación de cada técnica, quisimos obtener los resultados de cada técnica utilizando características y umbrales diferentes. Para ello creamos tres métodos, uno por cada técnica, que realizan una batería de clasificaciones para todas las combinaciones posibles de características y umbrales. Los umbrales de cada característica fueron previamente establecidos según sus buenos resultados en pruebas previas. Estos tres métodos pertenecen a la clase `Evaluacion` del paquete `evaluacion`.

Para el almacenamiento de los todos los resultados obtenidos, decidimos crear una clase `Tabla`. Para cada técnica, creamos tantas objetos `Tabla` como combinaciones de características eran posibles, en este caso 31 objetos por cada técnica. Cada `Tabla` almacena, para una única combinación de características, todos los resultados de FRR y FAR para cada umbral utilizado.

Cada `Tabla` es posteriormente extraída a un fichero de texto plano cuyo nombre corresponde con la combinación de características que ha utilizado. Este proceso se realiza mediante el método `generacion_txt`() de la clase `Extraccion`.

Una vez comprobado que calculábamos los todos los resultados y los almacenábamos correctamente, generamos gráficas para cada fichero de resultados obtenido. Para ello, utilizamos el programa *gnuplot* [16], que, de manera sencilla, nos permitió crear todas las gráficas que deseábamos. Todas ellas están disponibles en los anexos 1, 2 y 3.

En estas gráficas, somos capaces de observar dónde se encuentra el ERR para cada combinación de características. Sin embargo, mediante ellas no podemos determinar su valor con exactitud. Por este motivo, decidimos implementar uno nuevo método, el cual calcula la ERR mediante los resultados de FRR y FAR almacenados en un fichero de texto concreto. El proceso

para realizar este cálculo se detalla en la sección 3.3. Mediante este nuevo método, generamos una `Tabla` para cada técnica, la cual almacena las ERR de cada combinación de características utilizadas, junto con los umbrales con los que se alcanzan. Finalmente, las tres `Tablas` son extraídas en ficheros de texto plano. Las ERR de cada técnica están disponibles en los anexos 4, 5 y 6.

# 5. Evaluación de las distintas técnicas con distintos parámetros

En esta sección, vamos a hacer un análisis de todos los resultados obtenidos por cada una de las técnicas que hemos implementado. Este análisis nos permitirá responder a todas las preguntas que nos hemos ido planteando a lo largo del proyecto.

Como se explica en la sección 3.3, cuanto menores sean los tres tipos de tasas de error, mejor rendimiento y mayor seguridad tendrá el sistema. Por tanto, encontraremos las mejores combinaciones para cada técnica donde se minimizan los valores del ERR, ya que esta tasa indica el momento en el que FRR y FAR están equilibrados.

Vamos a dividir el análisis en cuatro partes, tres correspondientes al análisis individual de cada una de las técnicas y una cuarta, que corresponde a una comparación entre los resultados de los tres primeros.

## 5.1. Análisis del sistema de autenticación MAN

Antes de comenzar a analizar esta técnica, vamos a explicar la notación utilizada en todas sus gráficas, para entender fácilmente sus resultados y estos nos ayuden a analizarla correctamente. Cada gráfica corresponde a una combinación concreta de características, donde:

- El eje horizontal comprende cada una de las posibles combinaciones de umbrales para el conjunto de características actual.
- El eje vertical comprende todos los valores posibles del FRR y FAR, es decir, entre 0 y 1.
- Y para cada combinación de umbrales, observamos dos barras de colores, una para su valor de FRR (verde) y otra para su valor de FAR (roja).

Decidimos utilizar este tipo de representación debido a la multitud de umbrales que tiene esta técnica (uno por cada característica). Si tuviésemos un único umbral (como ocurre con las otras dos técnicas), podríamos dibujar cada gráfica como una función lineal en la que el umbral va cambiando. Sin embargo, con cinco umbrales, tendríamos una representación con múltiples dimensiones que sería muy confusa para el lector.

Como consecuencia de este mismo problema, nos encontramos con que si el conjunto de valores distintos de cada umbral es muy grande, las gráficas quedan ininteligibles. Esto nos llevó a decidir dibujarlas utilizando únicamente dos valores para cada umbral, dejándonos en el peor de los casos con 32 combinaciones de umbrales diferentes. Sin embargo, para aumentar la efectividad en el cálculo de los ERR, utilizamos cinco valores distintos para cada umbral. Todas las gráficas de esta técnica están disponibles en el anexo 1.

Una vez aclarado la notación y el tipo de representación de cada gráfica, podemos comenzar con el análisis de la técnica.

Lo más importante es "¿Qué combinación de características ofrece mejor ERR?". En este caso, la mejor combinación es (DT, FT1), obteniendo un valor de ERR = 0.13884804. Sin embargo, es seguida muy de cerca de por (DT, FT4) con ERR = 0.13997549 y (DT, FT3) con ERR = 0.14020097. Este resultado se consigue con los siguientes valores de umbral: $U_{DT}$= 1.5 y $U_{FT1}$= 1.75.

Otro dato curioso es la gran capacidad de decisión del *dwell time* para esta técnica. En el momento en que descartamos esta característica, el rendimiento del sistema baja considerablemente. Por tanto, los mejores resultados se alcanzan utilizando esta característica.

Después de él, las características más determinantes son el FT1 y FT4. Las combinaciones que utilizan estas características tienen mejores resultados respecto a las demás.

También podemos ver cuáles son los mejores rendimientos según el número de características que utiliza:

1. ERR = 0.1463407 para la combinación (DT).
2. ERR = 0.13884804 para la combinación (DT, FT1).
3. ERR = 0.1489902 para la combinación (DT, FT1, FT2).
4. ERR = 0.14908333 para la combinación (DT, FT1, FT2, FT4).
5. ERR = 0.15180883 para la combinación (DT, FT1, FT2, FT3, FT4).

Otro dato de interés, son los valores más frecuentes de cada umbral a la hora de obtener la ERR:

- $U_{DT}$= 1.75
- $U_{FT1}$= 1.25
- $U_{FT2}$= 1.5
- $U_{FT3}$= 1.5
- $U_{FT4}$= 1.5

## 5.2. Análisis del sistema de autenticación MEANSTDV

Como en el anterior análisis, vamos a comenzar explicando la notación que siguen las gráficas de esta técnica, para facilitar el análisis y visualización de los resultados. Cada gráfica corresponde con una sola combinación de características donde:

- El eje horizontal corresponde con el número de desviaciones que se utilizan para calcular el umbral de la técnica. Este valor tiene el valor 0 como el más restrictivo y según va aumentando, el sistema es más permisivo.
- El eje vertical corresponde con el rango de valores posibles del FRR, FAR y ERR, es decir, entre 0 y 1.
- Y para cada número de desviaciones, la FRR se representa con una cruz verde, mientras que la FAR con un punto rojo.

En las gráficas, todos los puntos de cada tasa está unidos para que podamos observar mejor cómo van variando las dos tasas según aumenta el número desviaciones. Y además, podemos observar claramente la ERR en el punto donde el ambas se cortan. Todas las gráficas correspondientes a esta técnica están disponibles en el anexo 1.2.

El mejor rendimiento para esta técnica se consigue utilizando únicamente *dwell time*. Si observamos detenidamente todas las gráficas de esta técnica, podemos llegar a la misma conclusión. Utilizando solo esta técnica, el valor de ERR es igual a 0.14603677. Los valores del mismo, para el resto de combinaciones de características, están muy por encima, siendo la segunda mejor ERR = 0.1927402, utilizando (DT y FT2).

Igual que en la técnica anterior, también ocurre que el *dwell time* es la característica más determinante. Las combinaciones que lo incluyen tienen unos mejores resultados. Y además, se puede observar que siempre que utilizamos esta característica, cuanto mayor número de características utilicemos, el rendimiento suele ser mayor.

Algo también muy destacable es que todos las ERR se obtienen casi siempre en el mismo rango de desviaciones típicas, entre 0.25 y 0.50. La única combinación que incumple esto es curiosamente la que utiliza solo DT.

Una de las cosas más destacables es la evolución de cada una de las tasas en función del número de desviaciones:

- Siendo numDesv = 0, la FRR registra sus peores valores; y según numDesv va aumentando, la FRR va disminuyendo, consiguiendo de esta manera, mejores resultados. Esto significa que cuanto mayor sea el número de desviaciones típicas, menor número de usuarios genuinos serán rechazados.
- De forma contraria, la FAR obtiene sus mejores resultados cuando numDesv = 0; y va empeorando según aumenta. Esto quiere decir que cuanto mayor sea el número de desviaciones, mayor cantidad de impostores estaremos aceptando.

También podemos ver cuáles son los mejores rendimientos según el número de características que utiliza:

1. ERR = 0.14603677 para la combinación (DT).
2. ERR = 0.14603677 para la combinación (DT, FT2).
3. ERR = 0.20553431 para la combinación (DT, FT1, FT4).
4. ERR = 0.22120589 para la combinación (DT, FT1, FT2, FT3).
5. ERR = 0.2227206 para la combinación (DT, FT1, FT2, FT3, FT4).

## 5.3. Análisis del sistema de autenticación ZSCORE

Igual que en las técnicas anteriores, vamos a comenzar explicando la notación de las gráficas de ZSCORE, las cuales nos van a ayudar enormemente a entender los resultados de forma sencilla. Cada gráfica corresponde a una sola combinación de características:

- El eje X corresponde al porcentaje de aciertos mínimos que necesita una entrada para ser aceptada. Como es de esperar, varía desde el 0% al 100%.
- El eje Y corresponde a los posibles valores de los tres tipos de tasas de error, que fluctúan entre 0 y 1.
- Y para cada porcentaje de aciertos, la FRR se representa con una cruz verde y la FAR con un punto rojo.

Todos y cada uno de los puntos del FRR y FAR están unidos para ayudarnos a imaginar cuáles serían sus respectivos valores del porcentaje de aciertos que no hemos utilizado. Sin embargo, al utilizar números enteros para estos porcentajes, el paso de un porcentaje a otro a veces es representado con un salto. Todas las gráficas pertenecientes a esta técnica pueden verse en el anexo 1.3.

El mejor resultado se obtiene utilizando únicamente el *dwell time,* con la ERR = 0.14178921. Observando todas las gráficas de esta técnica, se visualiza perfectamente que el DT ofrece el mejor valor.

Como ocurre en las otras dos técnicas, el *dwell time* es la característica más influyente a la hora de mejorar el rendimiento del sistema. Si la eliminamos de nuestras combinaciones de características, la ERR del sistema aumenta muchísimo. Y además, se produce un fenómeno muy curioso con ella:

- Si incluimos *dwell time*, el valor del ERR es directamente proporcional al número de características que utilicemos. Esto quiere decir que cuantas más características utilicemos mayor será el valor de ERR, es decir, peor será el rendimiento del sistema.
- Si no consideramos *dwell time*, ocurre lo contrario y el valor de ERR es inversamente proporcional al número de características que utilicemos. Lo que significa, que cuantas más características utilicemos, mejor será el rendimiento del sistema.

Otro dato concluyente es que los mejores rendimientos de esta técnica siempre se dan en el rango de porcentaje de aciertos entre 82% y 96%.

También podemos destacar la evolución de las tasas FRR y FAR en función del porcentaje de aciertos. Si nos fijamos en las gráficas de esta técnica, veremos cómo según va aumentando el porcentaje de aciertos:

- La FRR va aumentando, aumentando, por tanto, el número de usuarios genuinos rechazados.
- Y la FAR disminuyendo, bajando, del mismo modo, el número de impostores rechazados.

A pesar de que tengamos calculadas las ERR de todas las combinaciones posibles, hay varias de ellas que realmente no permiten calcular esta tasa. Esto se debe a que su FRR y su FAR nunca llegarán a ser iguales. Esta conclusión se puede observar perfectamente en las gráficas de las siguientes combinaciones:

- FT1, FT2 (Figura 82)
- FT1, FT3 (Figura 84)
- FT1 (Figura 86)

- FT2, FT3 (Figura 88)
- FT2, FT4 (Figura 89)
- FT2 (Figura 90)
- FT3, FT4 (Figura 91)
- FT3 (Figura 92)
- FT4 (Figura 93)

También podemos observar cuáles son los mejores rendimientos según el número de características que utiliza:

1. ERR = 0.14178921 para la combinación (DT).
2. ERR = 0.15026961 para la combinación (DT, FT2).
3. ERR = 0.17085785 para la combinación (DT, FT1, FT2).
4. ERR = 0.1922108 para la combinación (DT, FT1, FT2, FT4).
5. ERR = 0.23301226 para la combinación (DT, FT1, FT2, FT3, FT4).

## 5.4. Comparación

A la hora de comparar las tres técnicas, nos permite responder a la gran pregunta de este trabajo: ¿Qué técnica es la mejor?. El orden según qué técnica tiene mejor rendimiento es:

1. Sistema de autenticación MAN (ERR = 0.13884804 con DT y FT1)
2. Sistema de autenticación ZSCORE (ERR = 0.14178921 con DT)
3. Sistema de autenticación MEANSTDV (ERR = 0.14603677 con DT)

Además, el MAN es mejor que las otras dos técnicas en cualquier combinación de características, exceptuando cuando utilizan solo d*well time*. En ese caso, se invierte el orden.

Sin embargo, las otras dos no mantienen el mismo orden, teniendo ZSCORE mejor rendimiento que MEANSTDV cuando ambas utilizan *dwell time*, y al revés cuando no lo hacen.

También podemos ver que técnica es mejor para sistemas críticos. Para ello, vamos a distinguir dos situaciones concretas:

1. Una FAR muy baja y una FRR aceptable. ***Ejemplo***: sistemas que protegen información de alta seguridad que prefieren rechazar usuarios genuinos en vez de que se les cuelen impostores.
2. Una FRR muy baja y una FAR aceptable. ***Ejemplo***: sistemas con mucho tránsito donde el FRR debe ser muy bajo para evitar tapones.

Si consideramos la noción “aceptable” a los valores menores que 0.3, en ambos casos la técnica más adecuada es MEANSTDV, con los siguientes valores:

1. Situación 1: FAR = 0.069416665 y FRR = 0.2992647.
2. Situación 2: FRR = 0.03627451 y FAR = 0.29338235.

# 6. Conclusiones y Trabajo Futuro

## 6.1. Conclusiones

En este trabajo, hemos conseguido implementar un sistema de autenticación de usuarios basado en dinámica del tecleo, el cual es capaz de utilizar tres técnicas de clasificación distintas. Esto fue posible gracias al estudio y la comprensión de cada una de ellas.

Además de eso, mediante una batería de entradas de prueba, hemos sido capaces responder todas las preguntas que nos planteamos al principio del proyecto. Entre todas ellas, destaca "¿Qué técnica ofrece los mejores resultados?" siendo la respuesta, el sistema de autenticación MAN. Esta técnica ha sido la que nos ha ofrecido un mejor rendimiento y eficacia, seguida por ZSCORE y MEANSTDV, respectivamente.

Otro de nuestros objetivos, era al menos alcanzar unos valores similares a los resultados en los artículos propios de cada técnica. Los resultados de cada uno fueron los siguientes:

- Sistema de autenticación MAN: FRR = 0.0145; FAR = 0.0189;
- Sistema de autenticación MEANSTDV: FRR = 0.01; FAR = 0.16;
- Sistema de autenticación ZSCORE: FRR = 0.02; FAR = 0.13;

En ninguno de ellos se valoró el rendimiento del sistema con ERR, sin embargo, tanto en MEANSTDV como en ZSCORE, hemos conseguido unos resultados similares.

También hemos visto cómo MEANSTDV es el mejor método de los tres para su uso en sistemas críticos.

Para concluir este trabajo, vamos a comentar nuestra opinión sobre la biometría de dinámica del tecleo. A lo largo de todo el proyecto, hemos ido viendo el gran potencial que tiene la dinámica del tecleo. Este tipo de biometría tiene una gran cantidad de ventajas, como son su bajo coste, su fácil uso, su capacidad de actuar sigilosamente o su no baja-nula intrusión en la privacidad de los usuarios. A pesar de que no exista una técnica que consiga un sistema totalmente seguro, este tipo de sistemas son muy útiles para complementar los sistemas de autenticación clásicos. Además, esto nos anima a seguir estudiando este campo tan prometedor, el cual, en la actualidad, no se limita a los teclados informáticos, sino que evoluciona día a día. Un ejemplo de ello es su aplicación en los teclados virtuales de los teléfonos móviles. Por tanto, concluimos en que la autenticación por dinámica de tecleo es el futuro inmediato para garantizar la mayor seguridad posible en los dispositivos informáticos.

## 6.2. Trabajo Futuro

Si el trabajo siguiese adelante, lo primero que tendríamos que hacer sería implementar las tres técnicas de aprendizaje automático que no nos han dado tiempo. Estas tres técnicas se basan en algoritmos concretos, todos ellos ya implementados en la librería Weka [17], la cual, nos facilita mucho las cosas. De esta manera, tendríamos que estudiar el proceso que sigue cada uno de ellos y el formato de datos que utilizan para poder ajustarnos a nuestras demandas. Según el artículo [3], estas tres técnicas son muy prometedoras, teniendo, en sus respectivos artículos, los siguientes resultados:

- K-means, Euclidian [9]: ERR = 0.038
- K-Nearest Neighbor [11]: ERR = 0.005
- Bayesian y Mínimo [13]: FRR = 0.081; FAR = 0.028

Siendo nuestra intención obtener, al menos, los mismos resultados.

Otro avance posible sería añadir el cálculo de tiempos totales y memoria utilizada que requiere una técnica en cada combinación. Esto nos permitirá valorar el sistema bajo esos aspectos, pudiendo llegar a descartar combinaciones que antes considerábamos buenas.

Concluido esto, nos sería muy útil implementar un sistema de captación de características biométricas por teclado. Esto nos permitirá recoger nuestros propios datos en bruto y generar tantos conjuntos de entradas de entradas como quisiéramos, inclusive, una biblioteca de datos biométricos de tecleo que cualquier sistema pudiese usar. La mayor complejidad que tiene esto es la captación de tiempos de pulsación y de vuelo de manera continua. Una vez tuviésemos eso, el resto no supondría un gran problema.

También podríamos ampliar nuestro estudio, por ejemplo, estudiando técnicas de texto libre, las cuales nos permitirían crear un sistema de autenticación continua. Estas técnicas son específicas para la autenticación de usuarios en periodos largos de tiempo, como una sesión en un sistema. Para su uso, tendríamos que poder recibir un flujo de datos en tiempo real y, al mismo tiempo, evaluar si corresponde con el usuario que ha iniciado sesión. Esto sería un gran reto por la complejidad de ambas cosas.

# 7. Conclusions and Future work

## 7.1. Conclusions

In this work, we have implemented a user authentication system based on keystroke dynamics, which is capable of using three different classification techniques. This was possible thanks to the study and understanding of each one of them.

In addition to that, through a battery of test entries, we were able to answer all the questions that we raised at the beginning of the project. Among them, we emphasize "What technique gives the best results?" whose answer is the MAN authentication system. This technique has been the one that has offered us better performance, followed by ZSCORE and MEANSTDV, respectively.

Another of our goals was to attain values similar to the results in the articles of each technique. The results of each were as follows:

- MAN authentication system: FRR = 0.0145; FAR = 0.0189;
- MEANSTDV authentication system: FRR = 0.01; FAR = 0.16;
- ZSCORE authentication system: FRR = 0.02; FAR = 0.13;

In none of them, the system performance was evaluated with ERR, however, in both MEANSTDV and ZSCORE, we have achieved similar results.

We have also stated that MEANSTDV is the best method of the three for use in critical systems.

To conclude, we will comment on our opinion on keystroke dynamics biometrics. Throughout the whole project, we have been seeing the great potential of the keystroke dynamics. This type of biometry has a lot of advantages, such as its low cost, its easy use, its ability to act quietly or its non-low intrusion into the user's privacy. Although there is no technique that can achieve a totally secure system, these types of systems are very useful to complement the classic authentication systems. In addition, this encourages us to continue studying this promising field, which, at present, is not limited to computer keyboards, but evolves day by day. An example of this is its application on the virtual keyboards of mobile phones. Therefore, we conclude that dynamic authentication using keystroke dynamics is the immediate future to ensure the highest possible security in computing devices.

## 7.2. Future work

If the work were to continue, the first thing we would have to do would be to implement the three automatic learning techniques that have not given us time. These three techniques are based on specific algorithms, all of them already implemented in the Weka library [17], which makes things much easier for us. In this way, we would have to study the process that each one follows and the format of data that they use to be able to adjust to our demands. According to [3], these three techniques are very promising, having, in their respective articles, the following results:

- K-means, Euclidian [9]: ERR = 0.038
- K-Nearest Neighbor [11]: ERR = 0.005
- Bayesian y Mínimo [13]: FRR = 0.081; FAR = 0.028

It is our intention to obtain, at least, the same results.

Another possible advance would be to add the calculation of total CPU time and memory used for each technique in each combination. This will allow us to evaluate the system under those aspects, being able to get to discard combinations that we not efficient enough.

Having concluded this, we would be very useful to implement a system of capturing biometric characteristics by keyboard. This will allow us to collect our own raw data and generate as many sets of input inputs as we would like, even a library of biometric typing data that any system could use. The greatest complexity of this line would be the capture of dwell and flight times continuously. Once we had that, the rest would not be a big problem.

We could also extend our study, for example, by studying free text techniques, which would allow us to create a continuous authentication system. These techniques are specific for user authentication over long periods of time, such as a session on a system. We should be able to receive a stream of data in real time and, at the same time, decide whether it corresponds to the user who is logged on or not. This would be a big challenge because of the complexity of both tasks.

# 8. Bibliografía

[1] Yu Zhong and Yunbin Deng. Recent Advances in User Authentication Using Keystroke Dynamics Biometrics, Volumen 2 de 6, capítulo 1. Gate to Computer Science and Research. DOI: 10.15579/gcsr.

[2] Definición de Biometría. https://es.wikipedia.org/wiki/Biometr%C3%ADa

[3] Pin Shen Teh, Andrew Beng Jin Teoh y Shigang Yue. A Survey of Keystroke Dynamics Biometrics. The Scientific World Journal, Volumen 2013 (2013), Article ID 408280, 24 pages. http://dx.doi.org/10.1155/2013/408280

[4] SYRIS Technology Corp. Technical Document About FAR, FRR and EER. 2004. http://documents.tips/documents/about-far-frr-eer.html

[5] Salil P. Banerjee y Damon L. Woodard. Biometric Authentication and Identification using Keystroke Dynamics: A Survey. Journal of Pattern Recognition Research, 7 2012. http://lsia.fi.uba.ar/papers/banerjee12.pdf

[6] Lívia C. F. Araújo, Luiz H. R. Sucupira Jr., Miguel G. Lizárraga, Lee L. Ling y João B. T. Yabu-Uti. User Authentication Through Typing Biometrics Features. IEEE Transactions on Signal Processing, Volumen 53, Numero 2, Febrero 2005.

[7] Rick Joyce y Gopal Gupta. Identity Authentication Based on Keystroke Latencies. Communications of the ACM, Febrero 1990, Volumen 33, Número 2.

[8] Sajjad Haider, Ahmed Abbas y Abbas K. Zaidi. A Multi-Technique Approach for User Identification through Keystroke Dynamics. Systems, Man, and Cybernetics, 2000 IEEE International Conference on. IEEE, DOI: 10.1109/ICSMC.2000.886039, páginas 1336-1341, Agosto 2002.

[9] Pilsung Kang, Seong-seob Hwang y Sungzoon Cho. Continual Retraining of Keystroke Dynamics Based Authenticator. S.-W. Lee and S.Z. Li, ICB 2007, LNCS 4642, páginas 1203–1211, 2007.

[10] https://es.wikipedia.org/wiki/K-means

[11] John C. Stewart, John V. Monaco, Sung-Hyuk Cha y Charles C. Tappert. An Investigation of Keystroke and Stylometry Traits for Authenticating Online Test Takers. 2011 International Joint Conference on Biometrics. IEEE, DOI: 10.1109/IJCB.2011.6117480, Diciembre 2011.

[12] https://es.wikipedia.org/wiki/K-vecinos_m%C3%A1s_cercanos

[13] Saleh Bleha, Charles Slivinsky y Bassam Hussien. Computer-Access Security Systems Using Keystroke Dynamics. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volumen 12, Número 12, Diciembre 1990.

[14] Conjunto de entradas. http://www.cs.cmu.edu/~keystroke/

[15] Definición Aprendizaje automático.
https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico

[16] gnuplot homepage. http://www.gnuplot.info/

[17] Weka. http://www.cs.waikato.ac.nz/ml/weka/

[18] K. Killourhy and R. Maxion. Why did my detector do that?!: predicting keystroke-dynamics error rates. 13th International Conference on Recent Advances in Intrusion Detection. Páginas 256–276, 2010.

# Anexos

## Anexo I: Gráficas MAN

**Figura 1: DT, FT1, FT2, FT3, FT4**

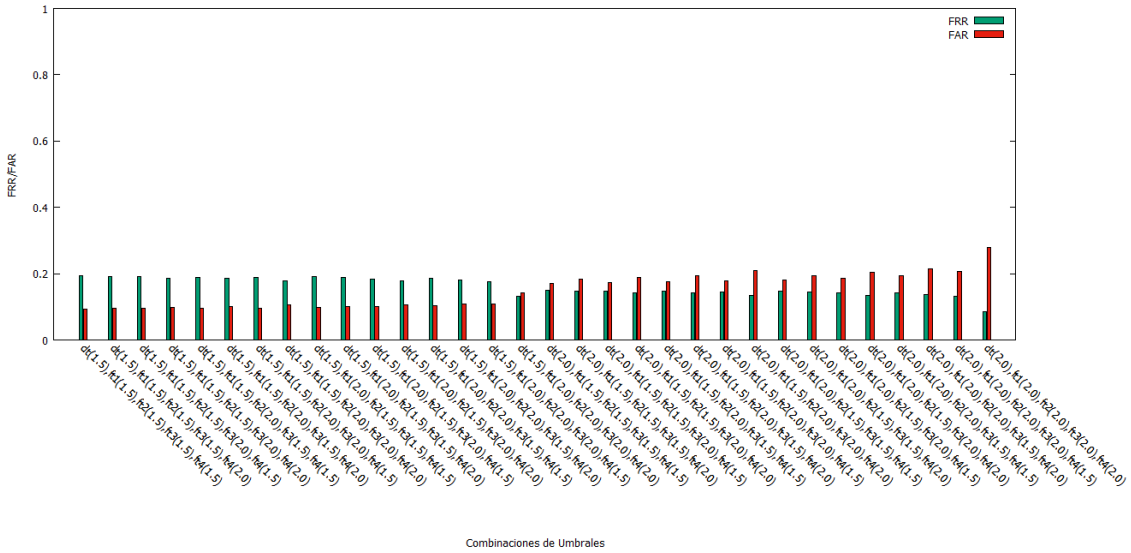



**Figura 2: DT, FT1, FT2, FT3**

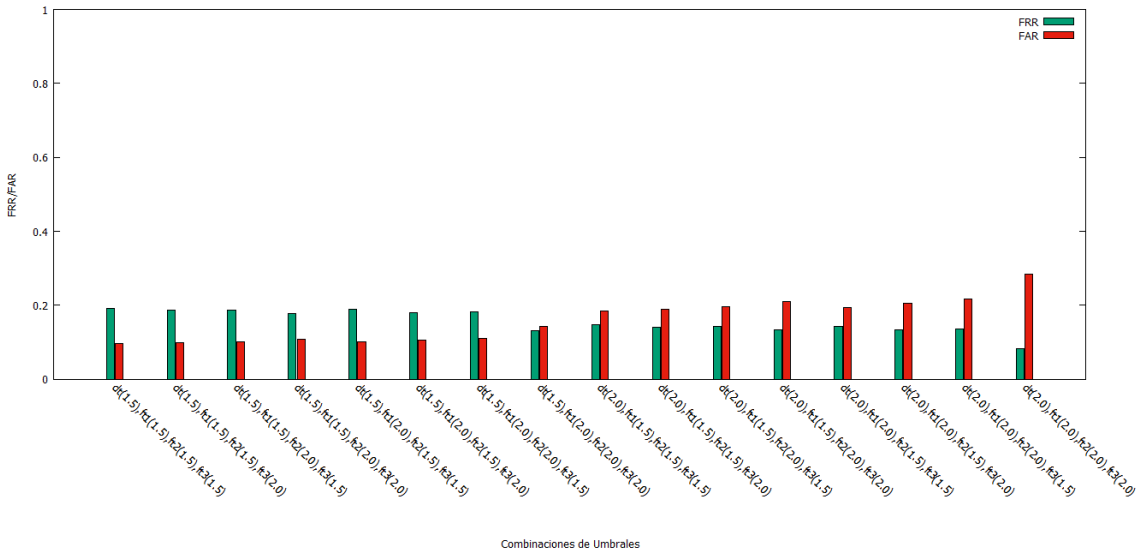


**Figura 3: DT, FT1, FT2, FT4**

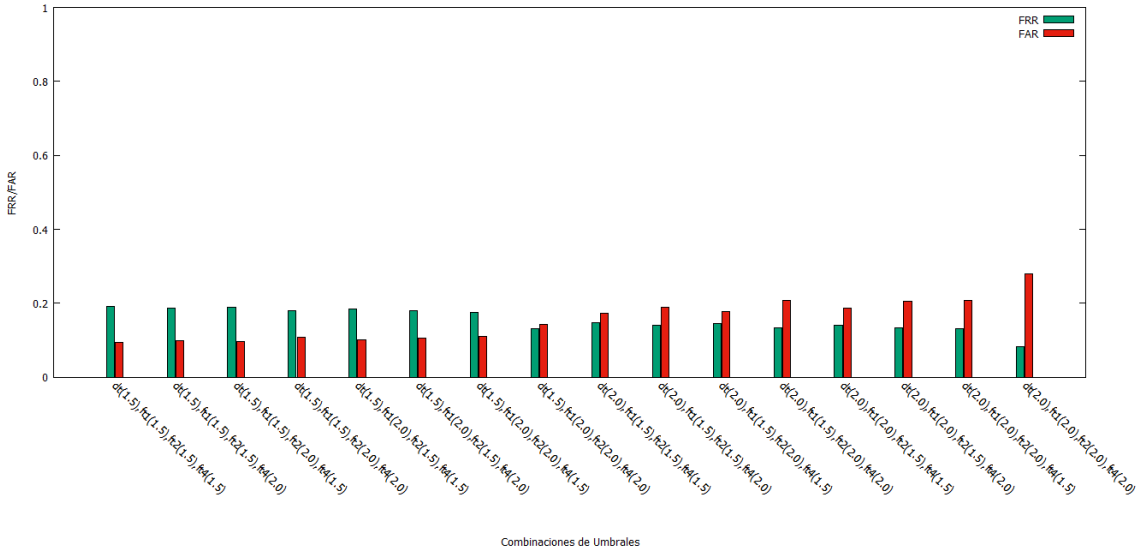



**Figura 4: DT, FT1, FT2**

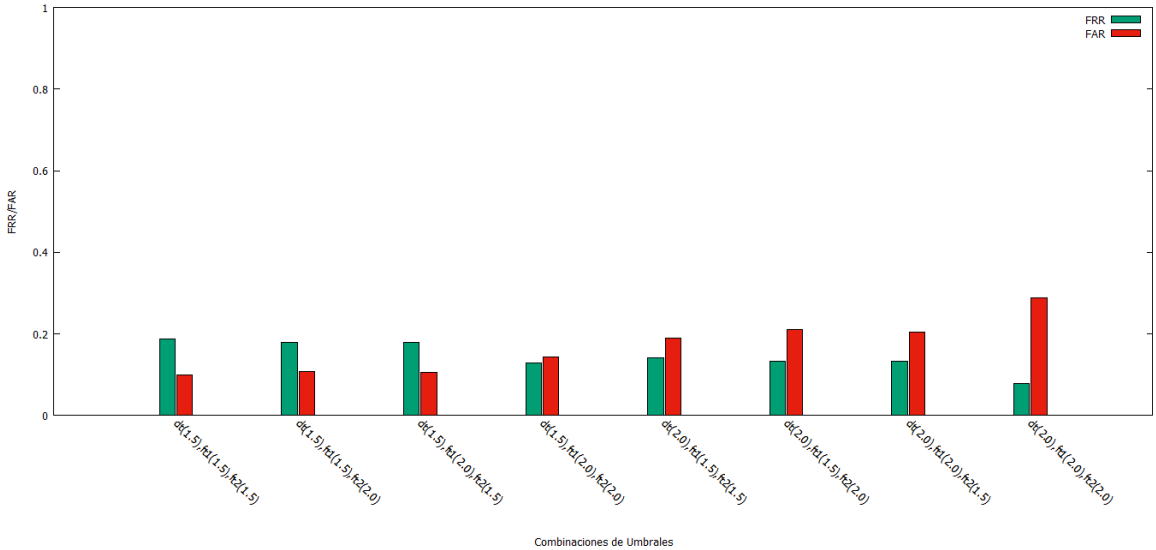



**Figura 5: DT, FT1, FT3, FT4**

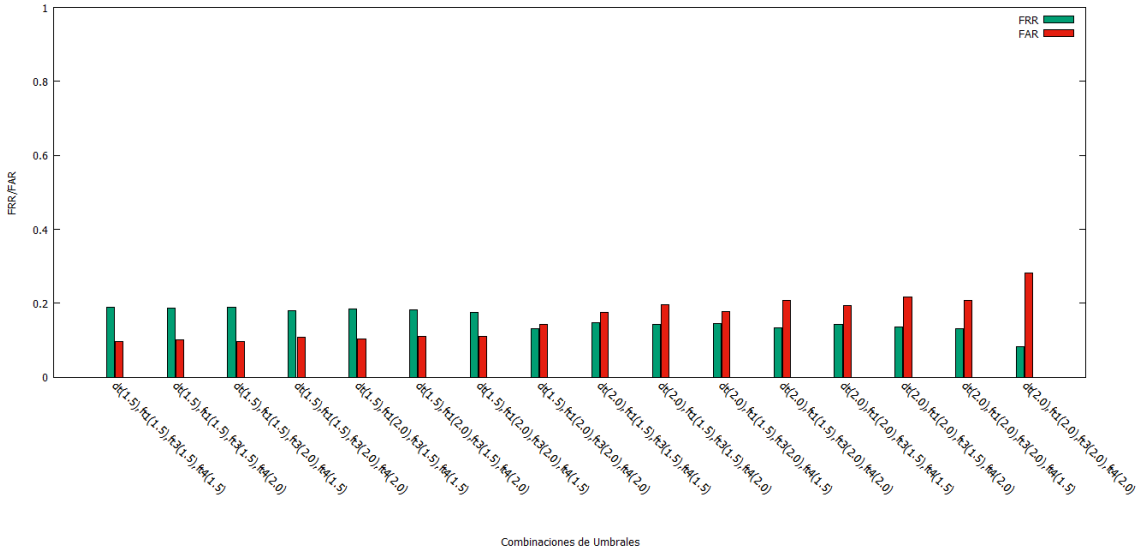


**Figura 6: DT, FT1, FT3**

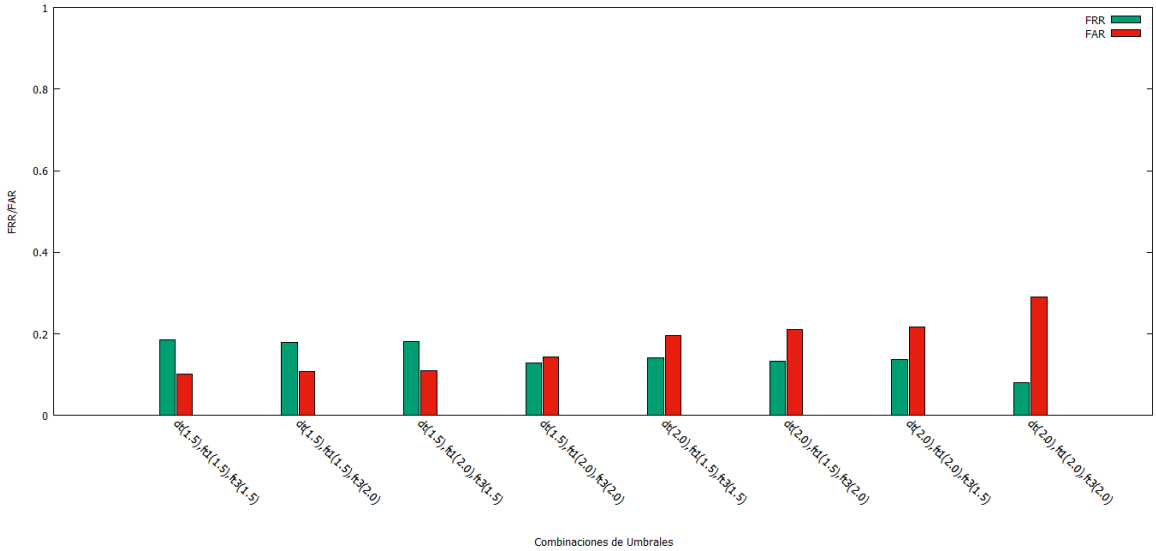



**Figura 7: DT, FT1, FT4**

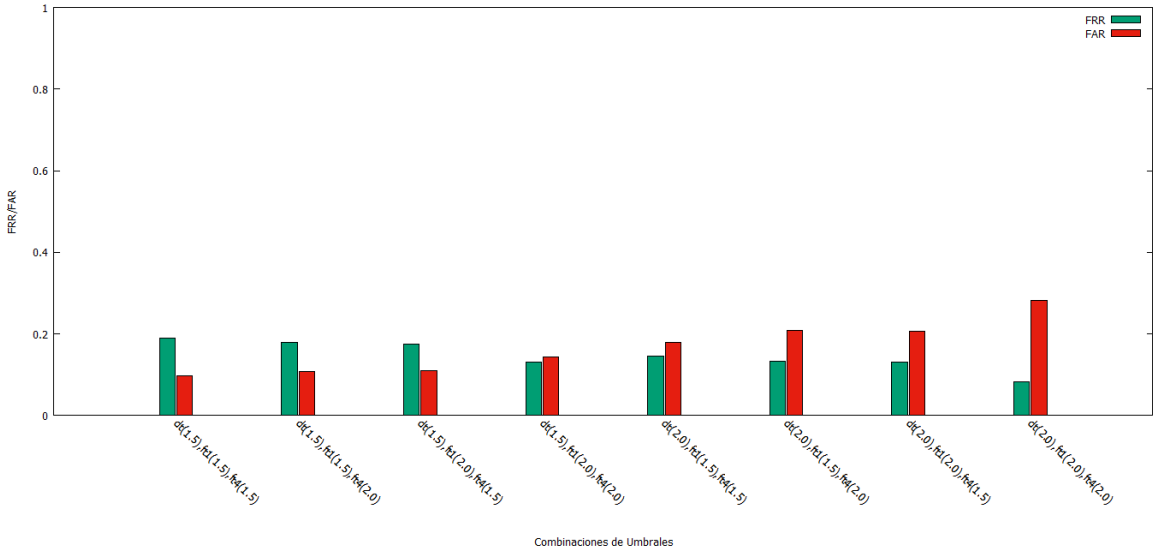



**Figura 8: DT, FT1**

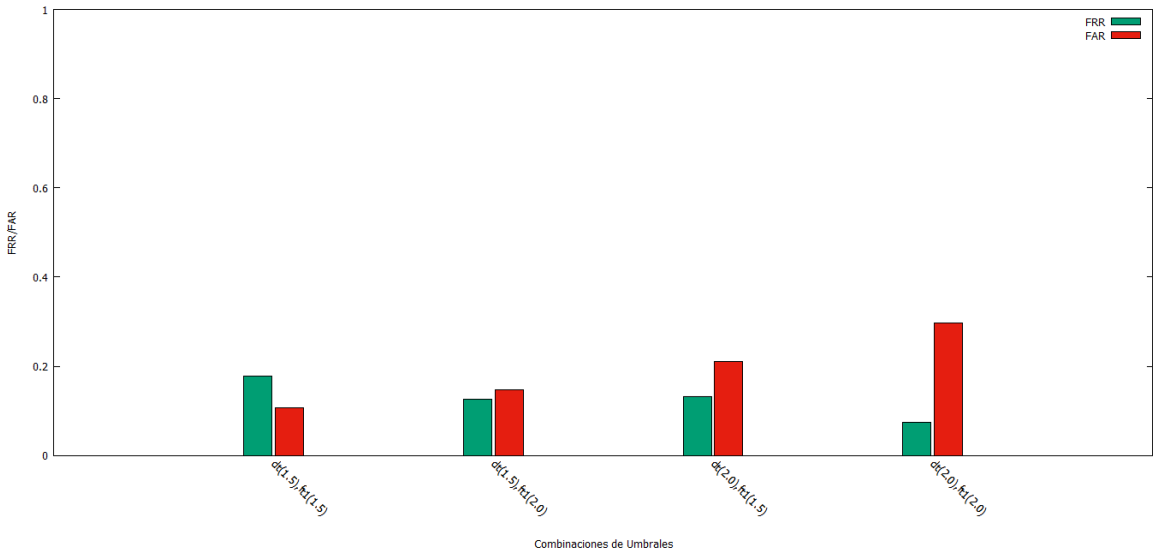


**Figura 9: DT, FT2, FT3, FT4**

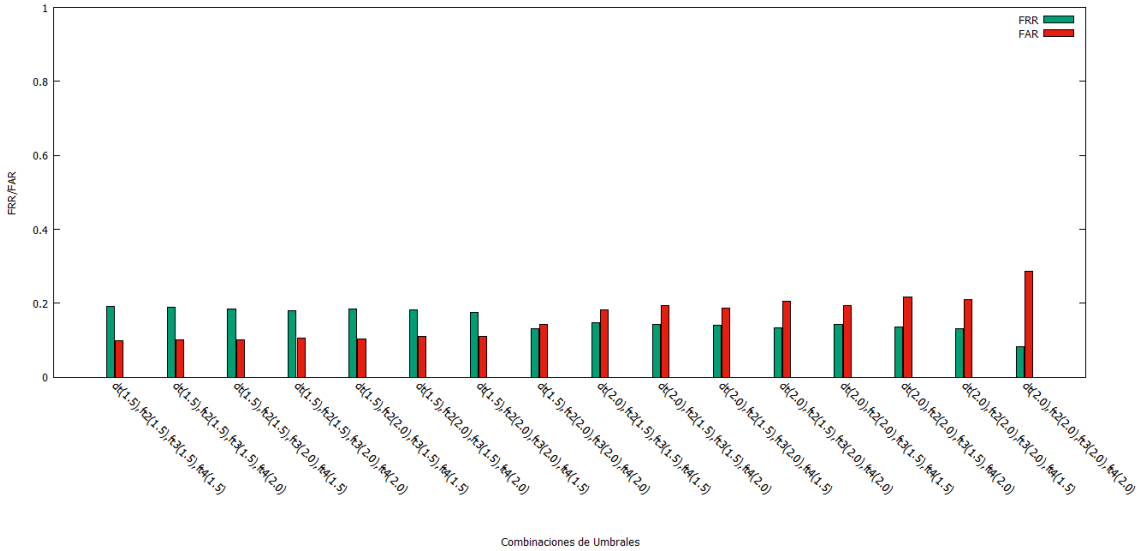



**Figura 10: DT, FT2, FT3**

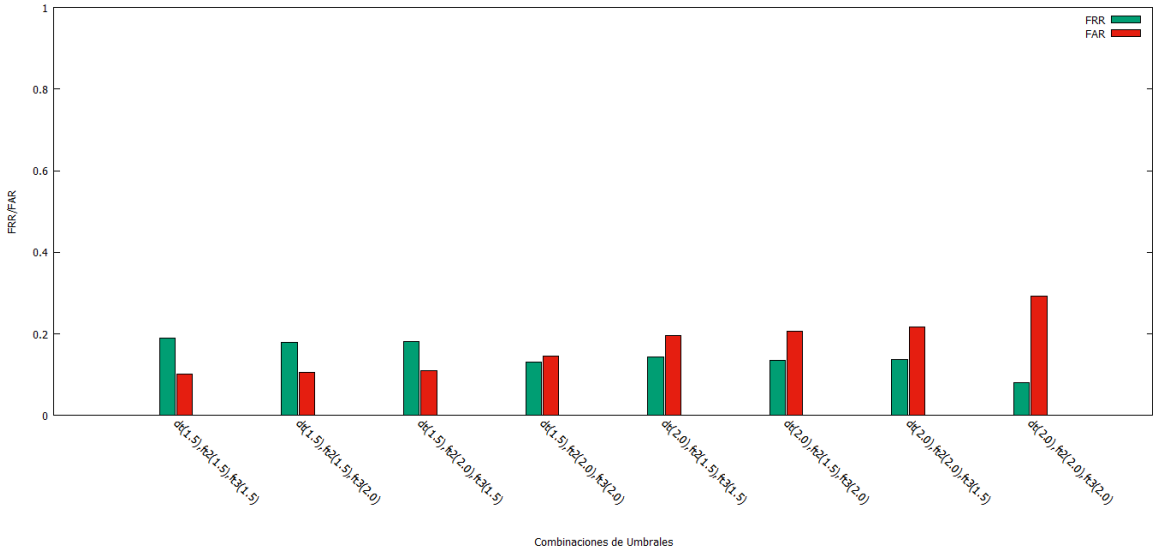



**Figura 11: DT, FT2, FT4**

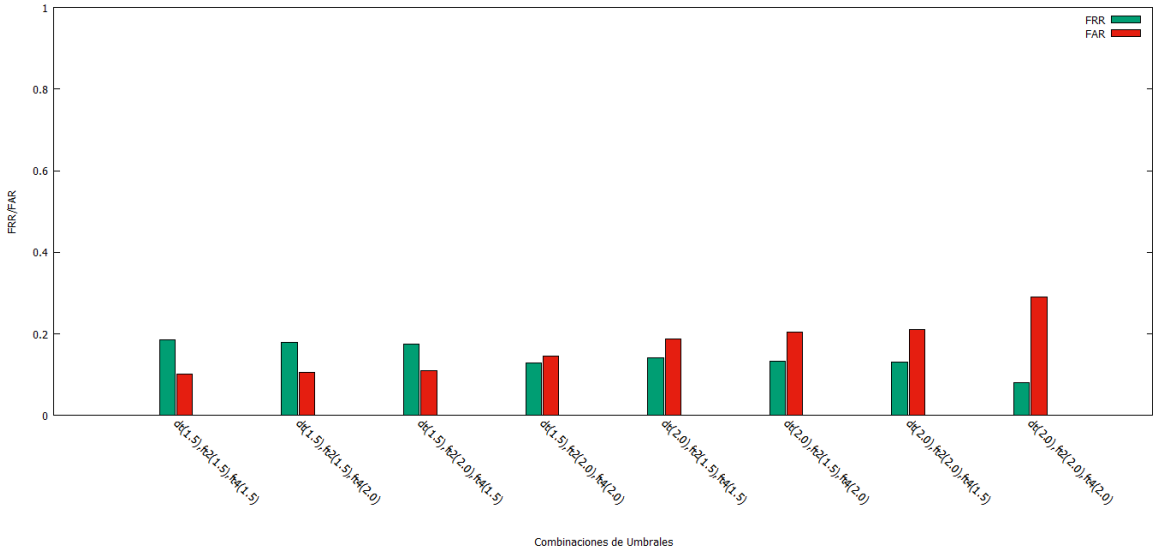


**Figura 12: DT, FT2**

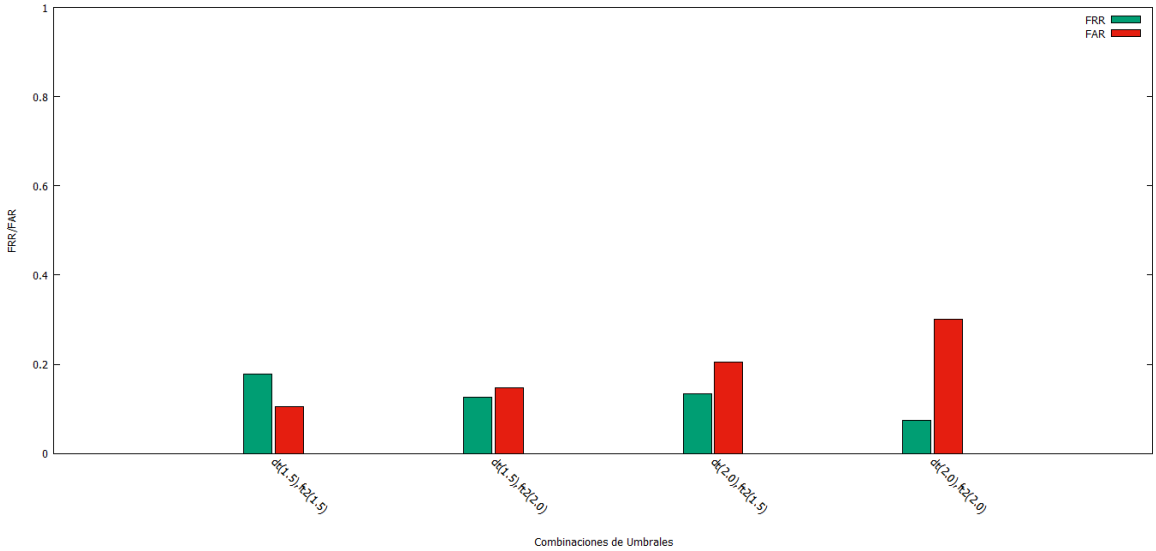



**Figura 13: DT, FT3, FT4**

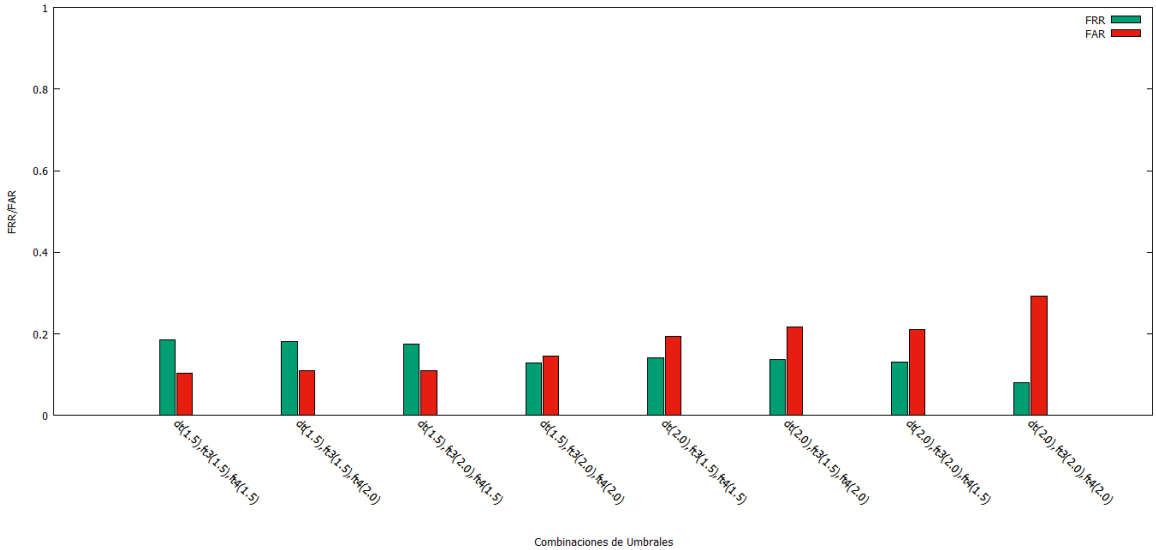



**Figura 14: DT, FT3**

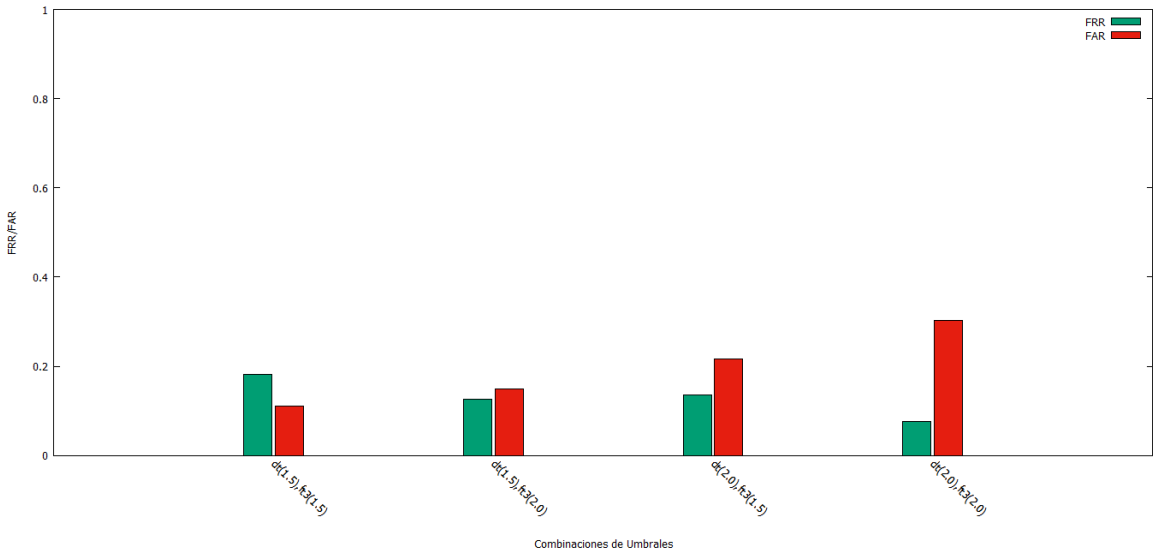


Figura 15: DT, FT4

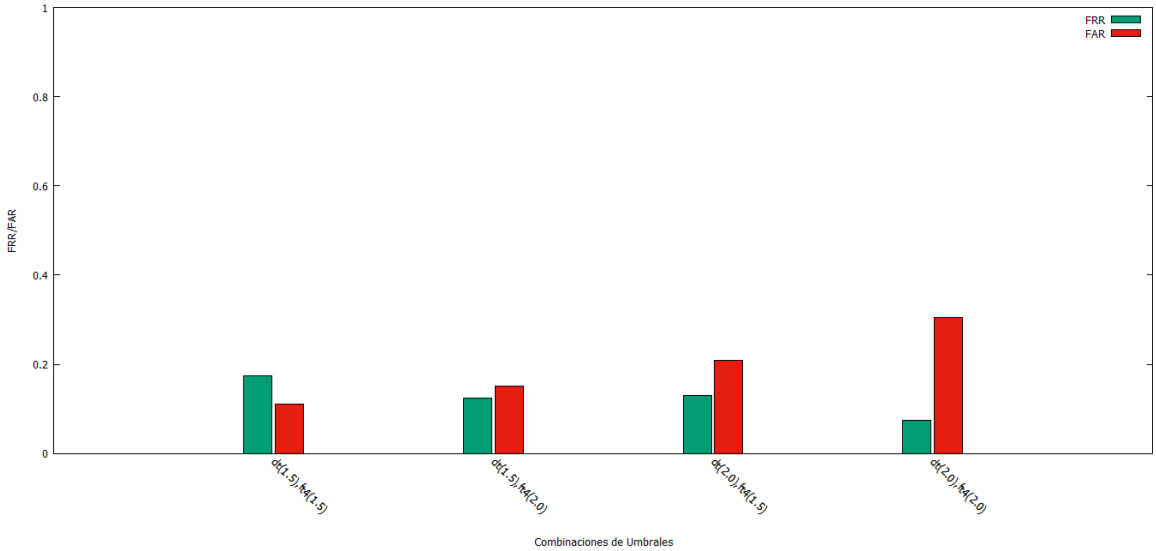



Figura 16: DT

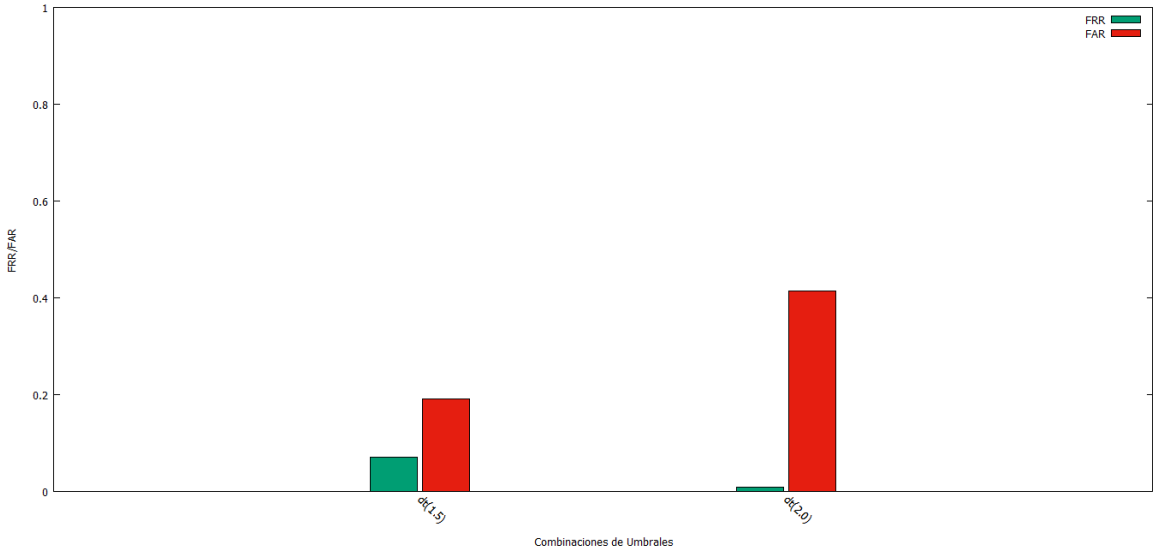



Figura 17: FT1, FT2, FT3, FT4

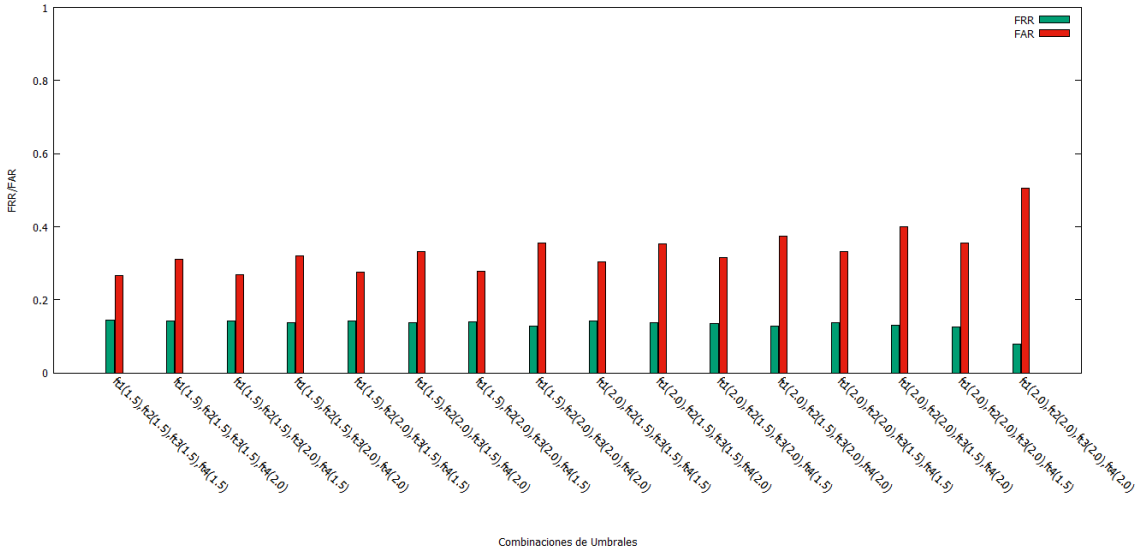


**Figura 18: FT1, FT2, FT3**

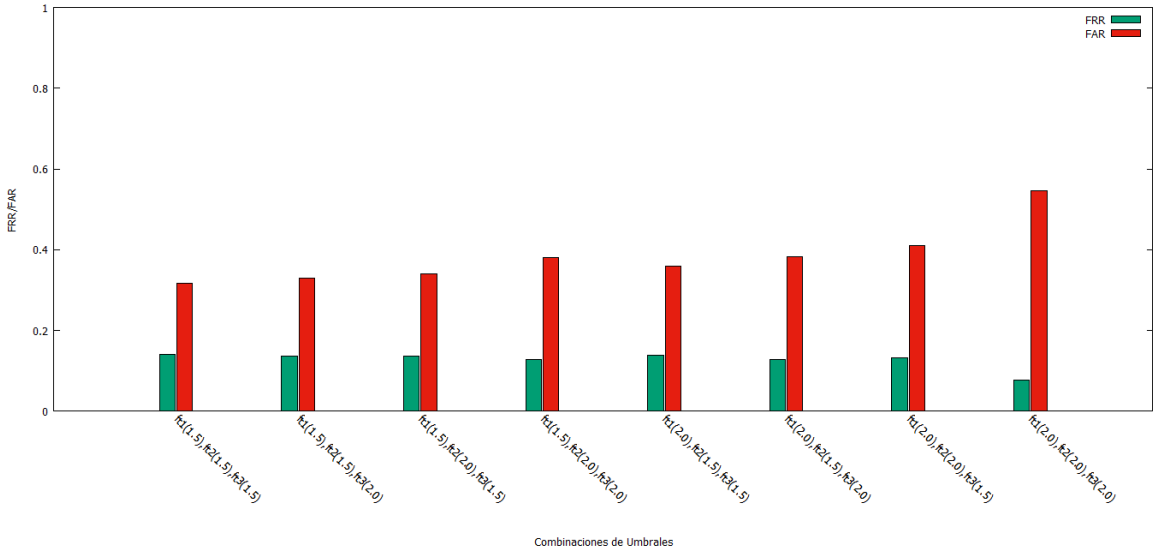



**Figura 19: FT1, FT2, FT4**

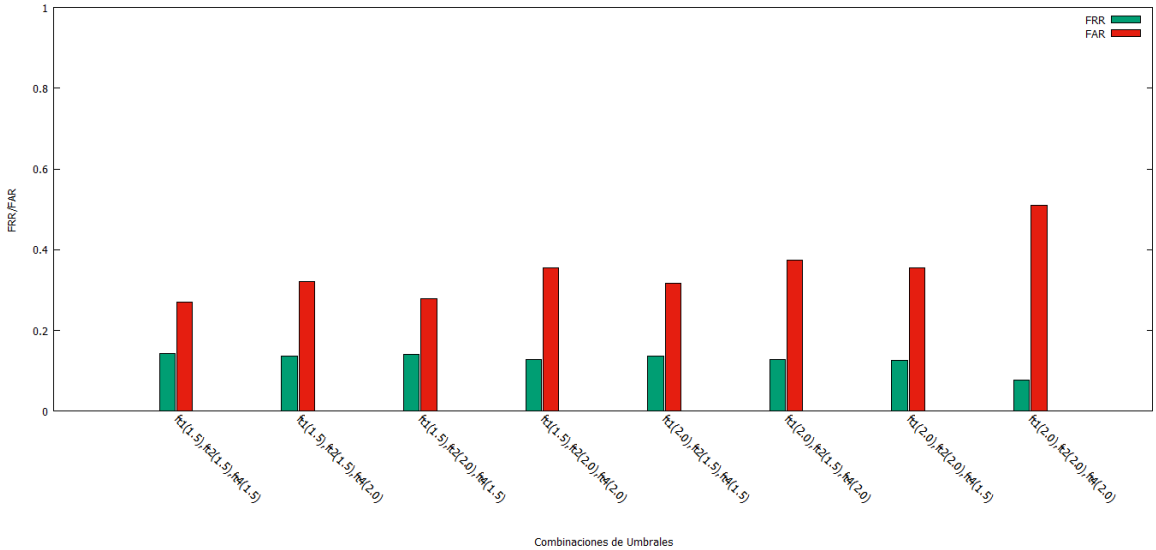



**Figura 20: FT1, FT2**

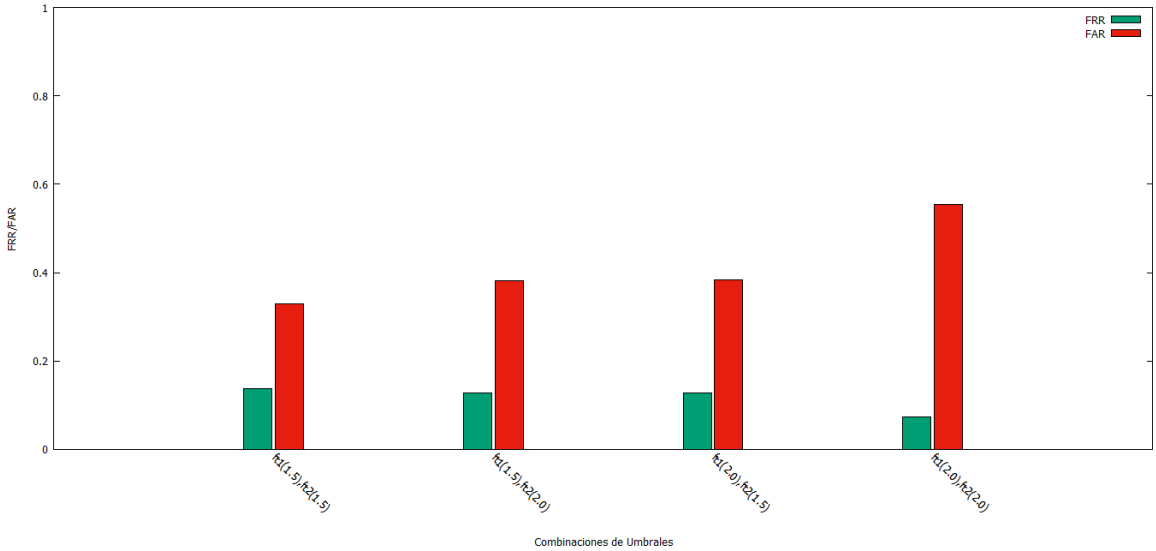


**Figura 21: FT1, FT3, FT4**

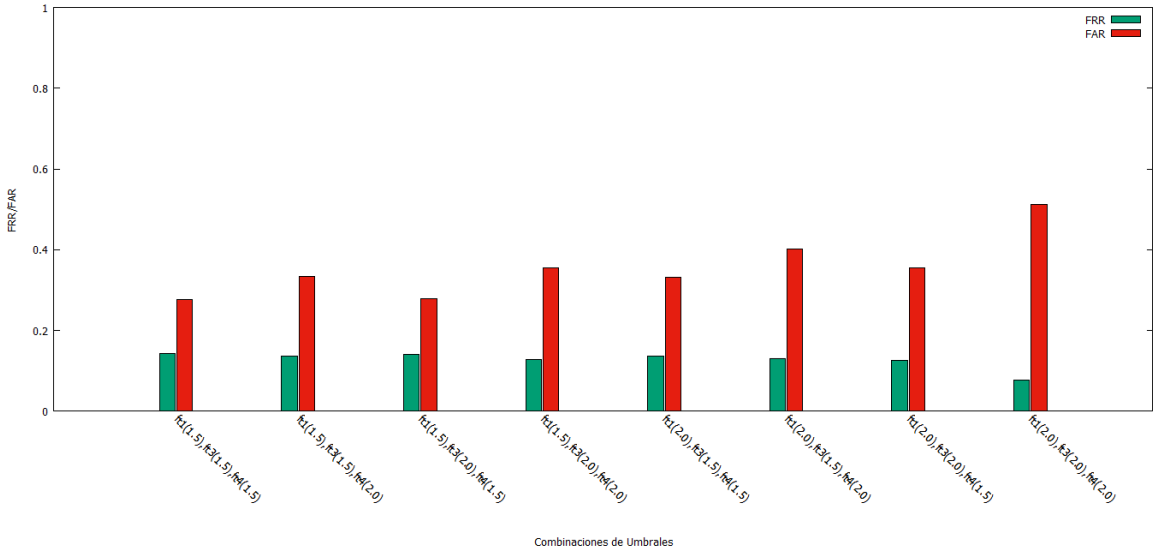



**Figura 22: FT1, FT3**

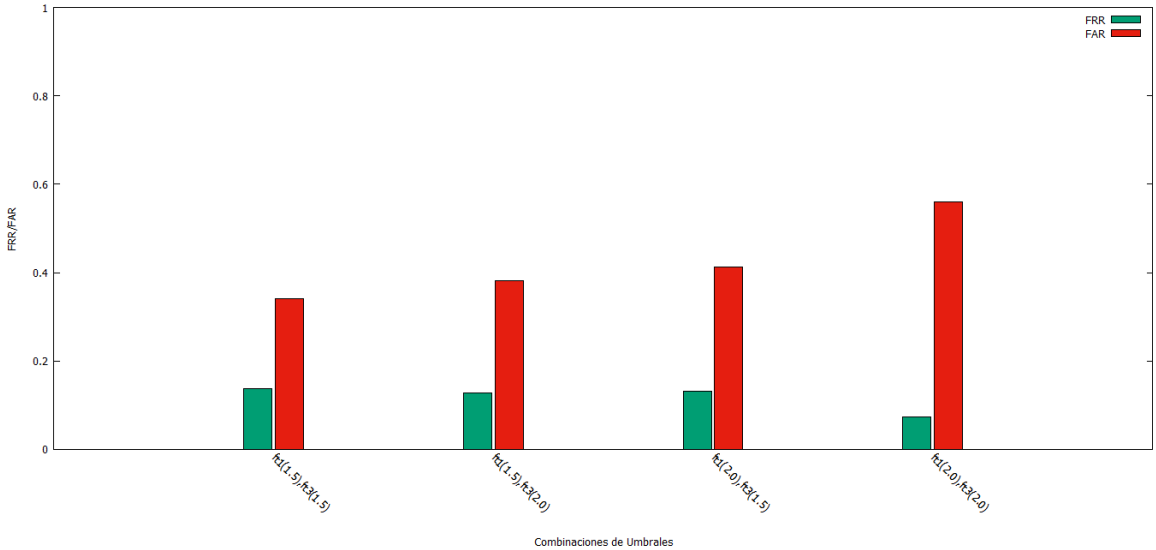



**Figura 23: FT1, FT4**

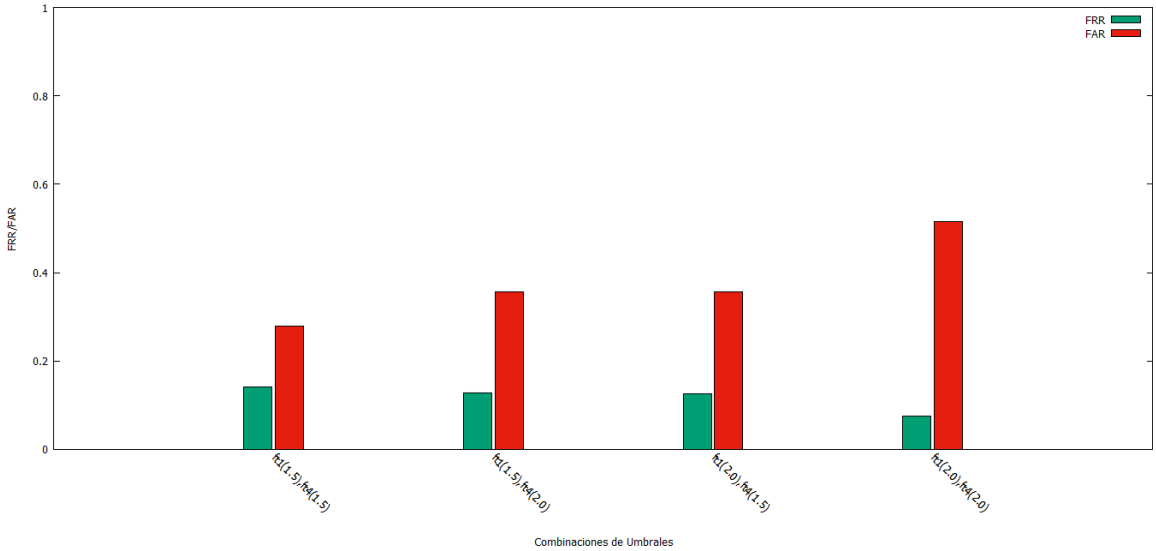


**Figura 24: FT1**

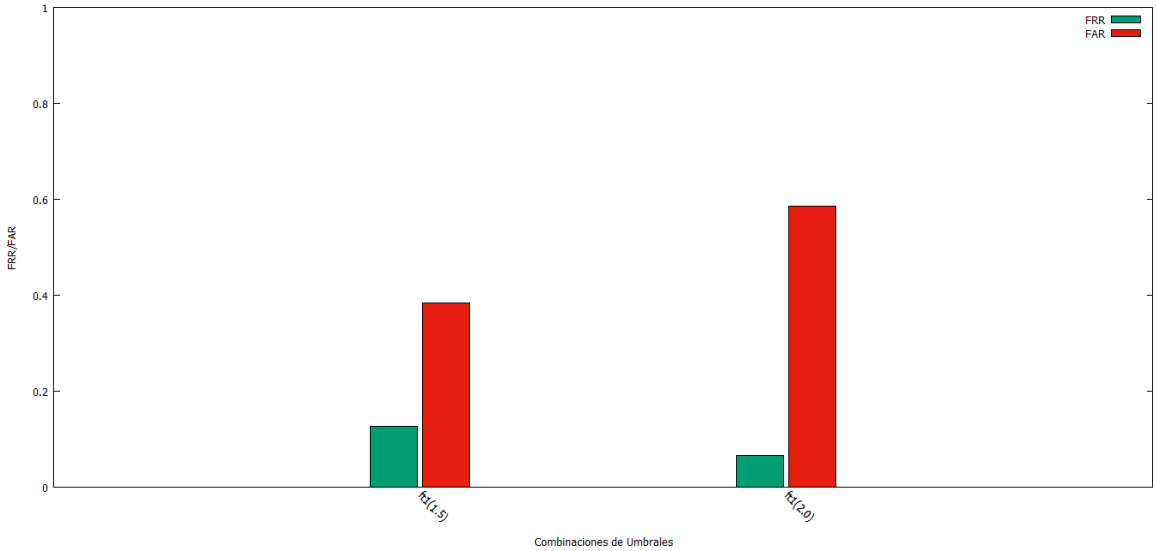



**Figura 25: FT2, FT3, FT4**

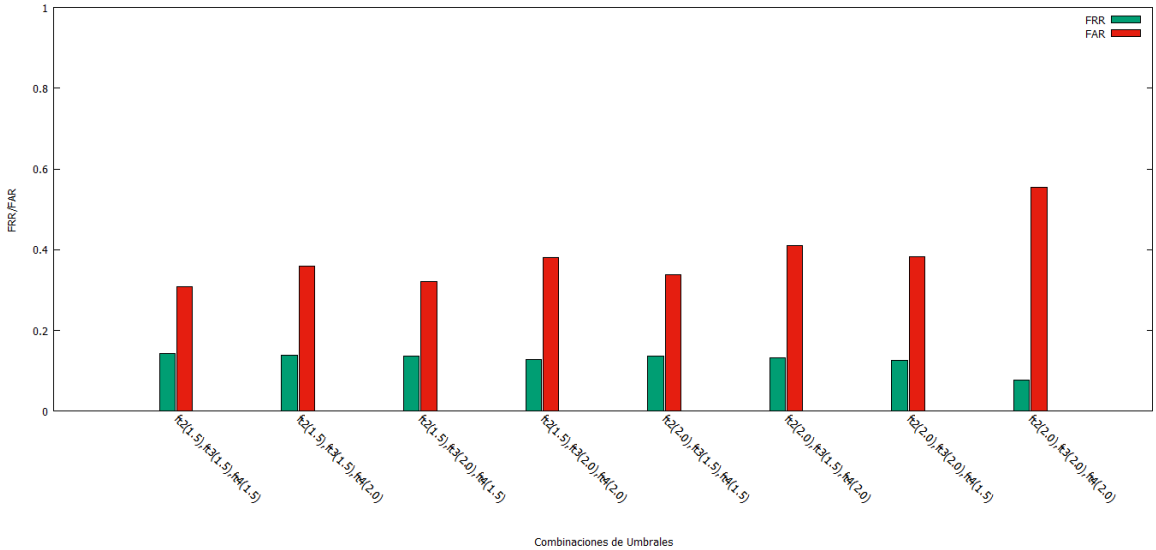



**Figura 26: FT2, FT3**

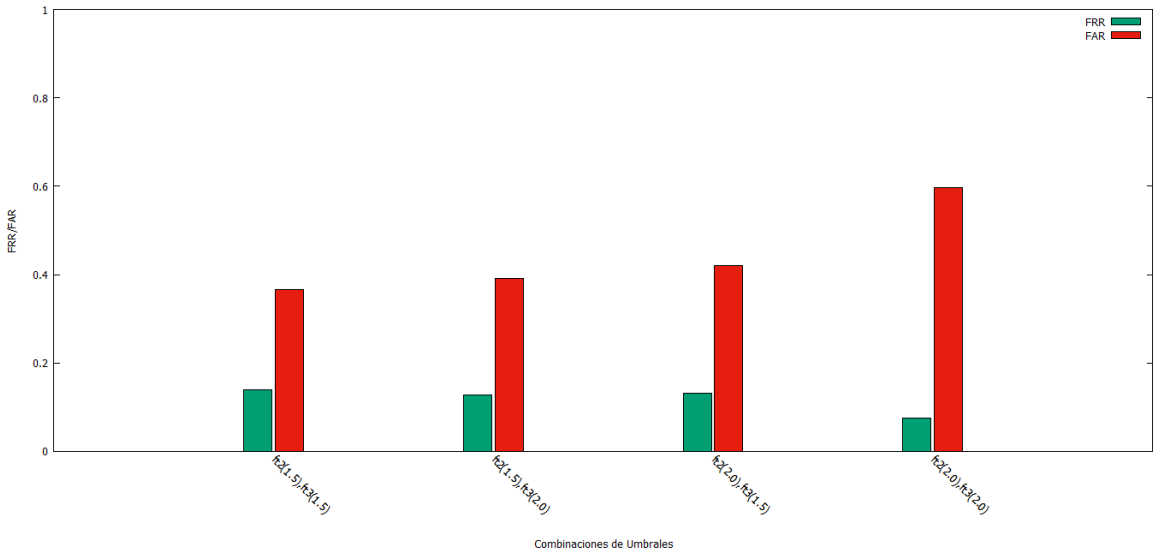


**Figura 27: FT2, FT4**

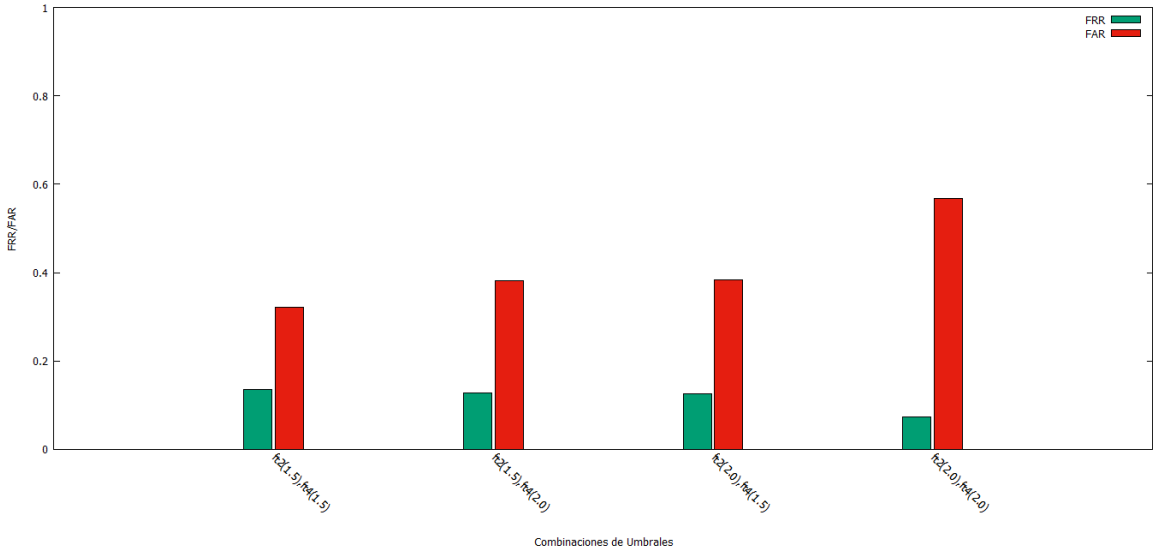



**Figura 28: FT2**

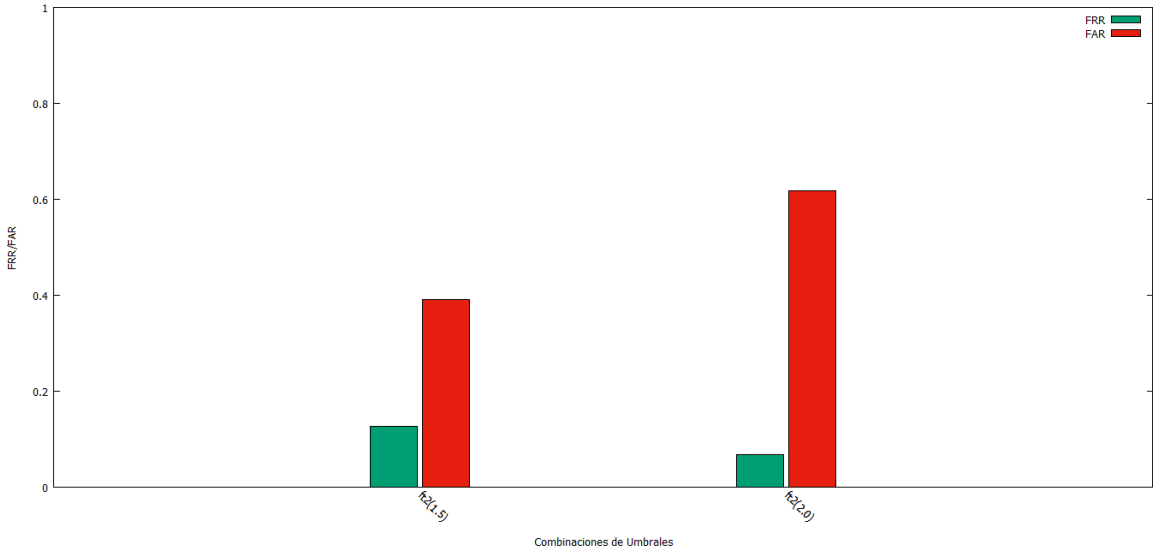



**Figura 29: FT3, FT4**

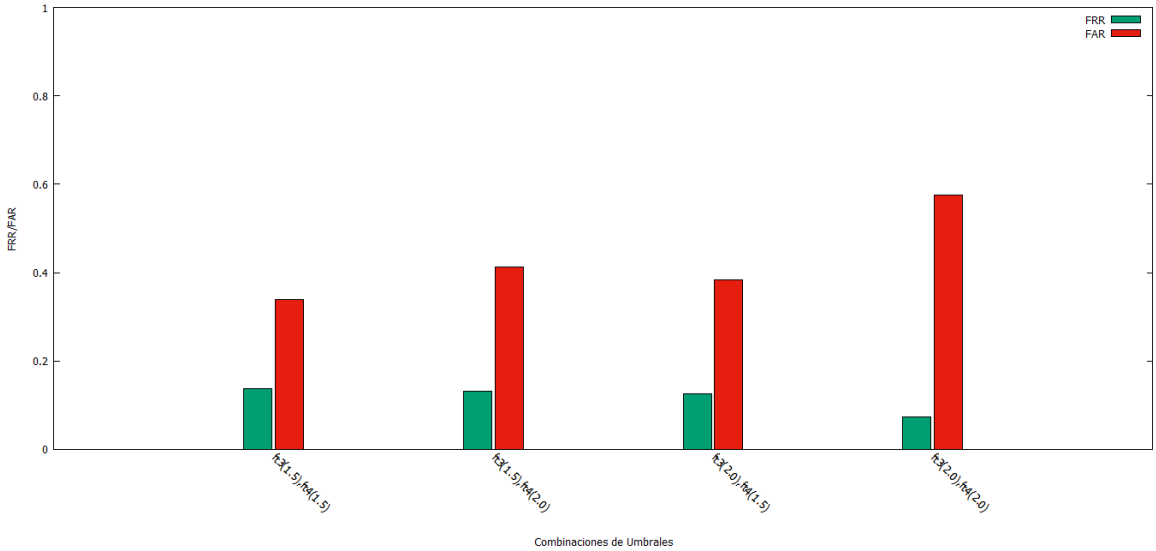


**Figura 30: FT3**

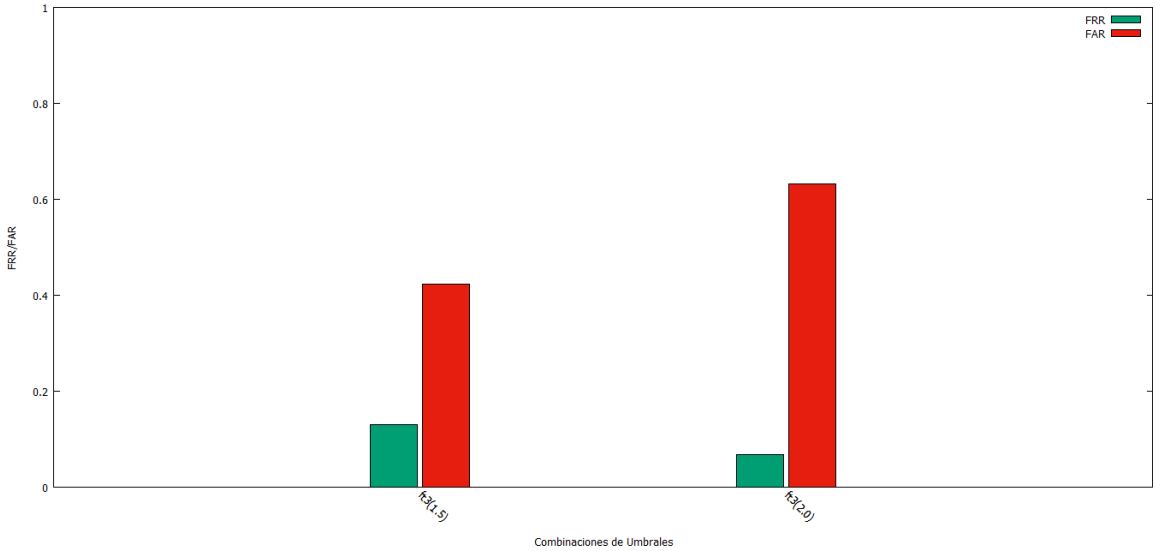
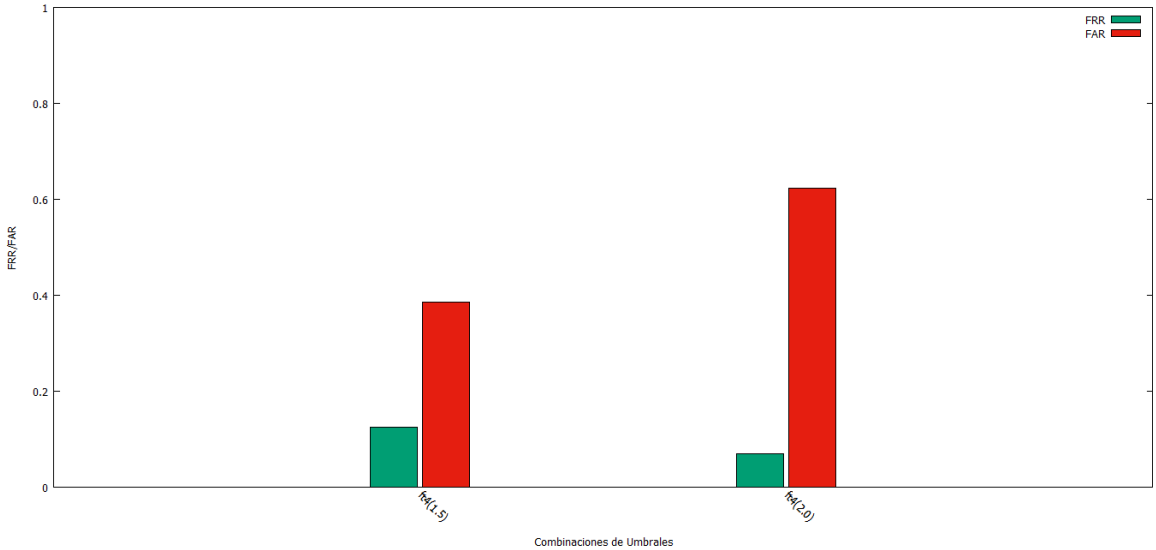



**Figura 31: FT4**



# Anexo II: Gráficas MEANSTDV

**Figura 32: DT, FT1, FT2, FT3, FT4**

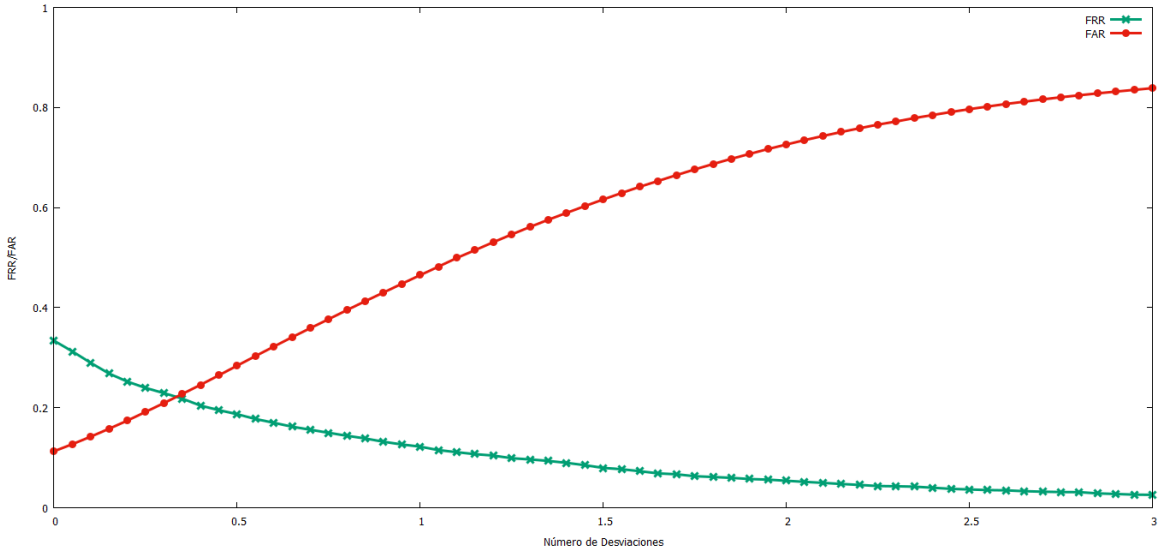



**Figura 33: DT, FT1, FT2, FT3**

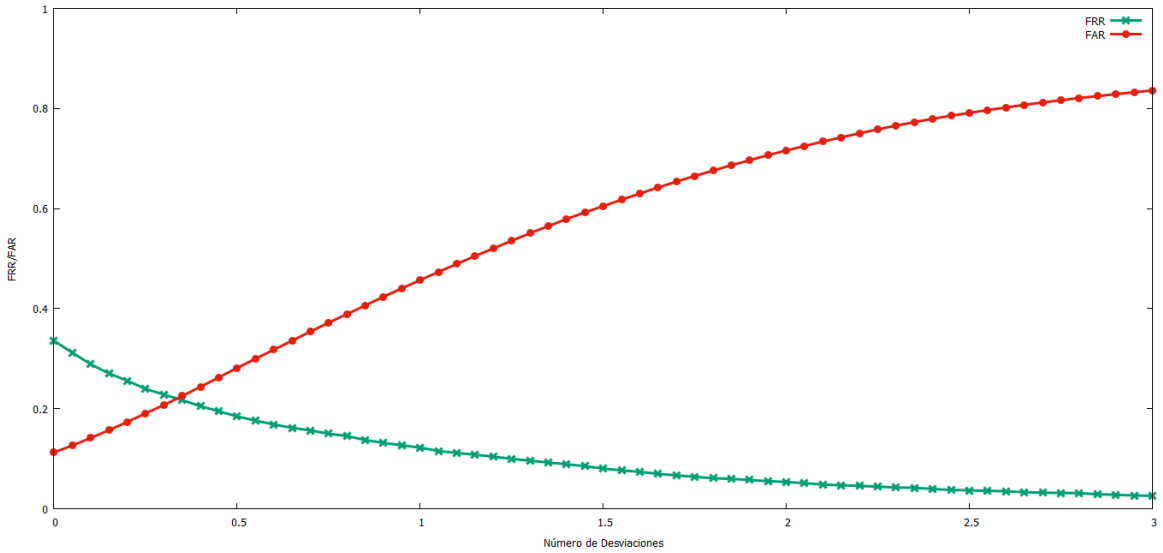


Figura 34: DT, FT1, FT2, FT4

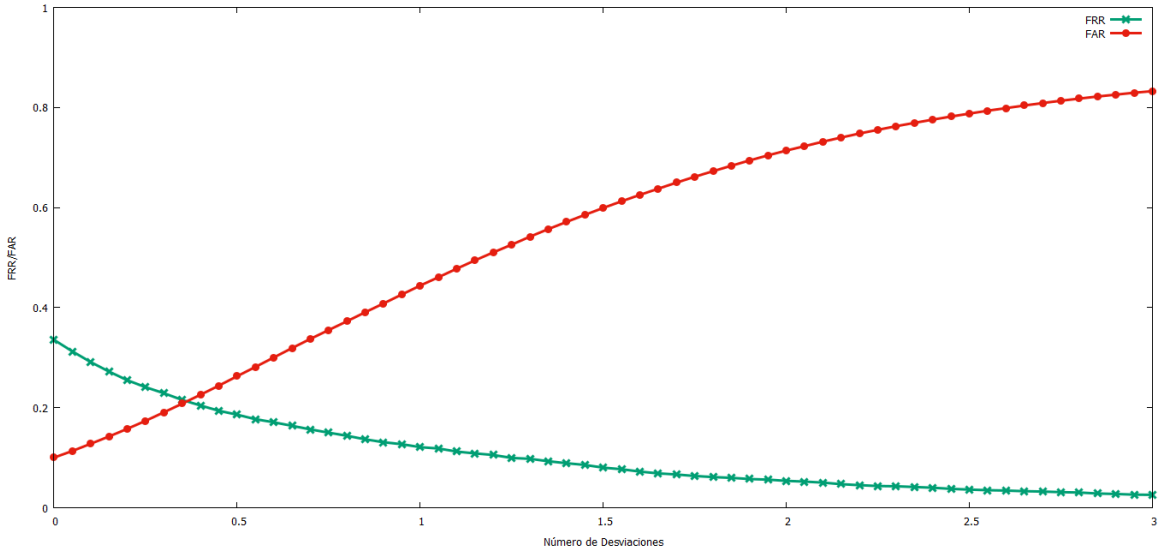



Figura 35: DT, FT1, FT2

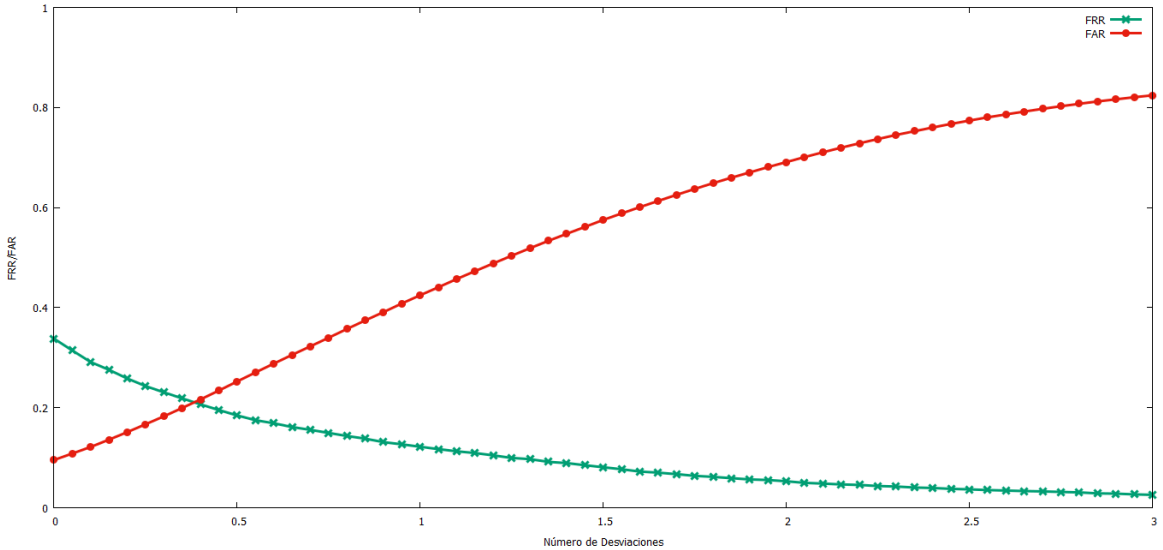



Figura 36: DT, FT1, FT3, FT4

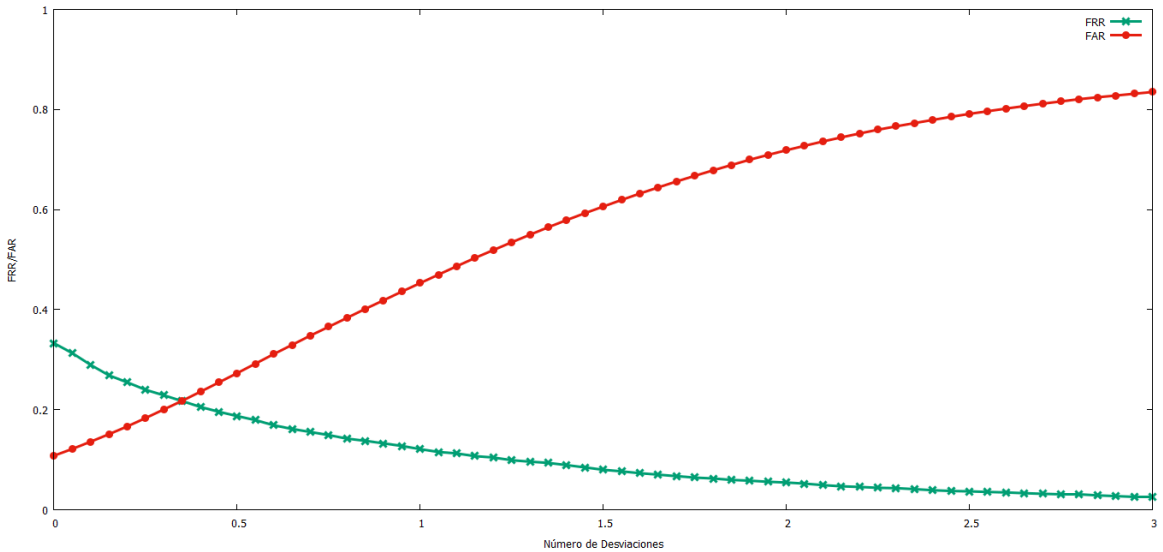


Figura 37: DT, FT1, FT3

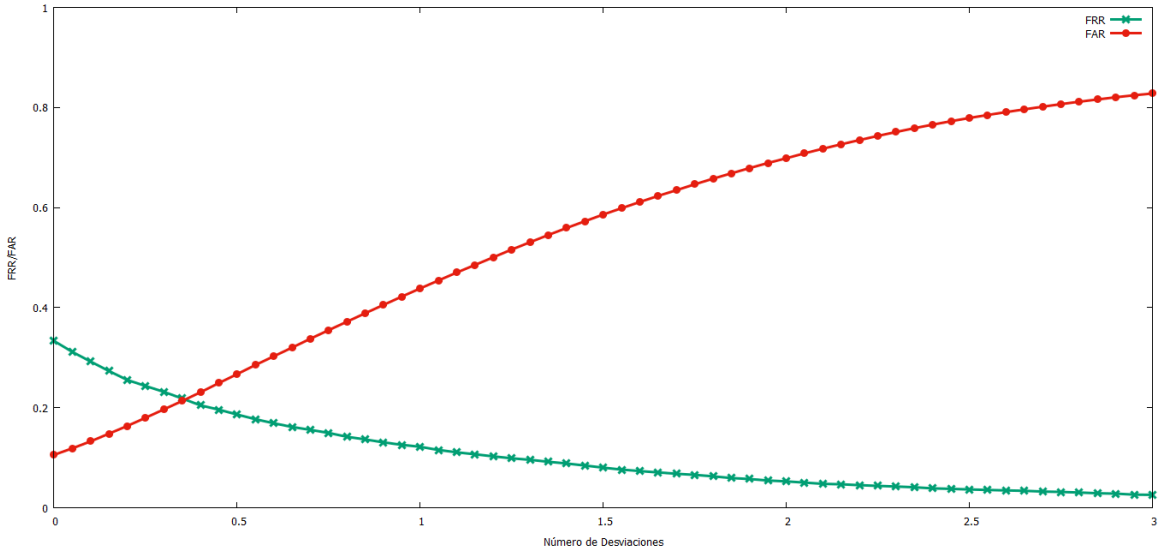



Figura 38: DT, FT1, FT4

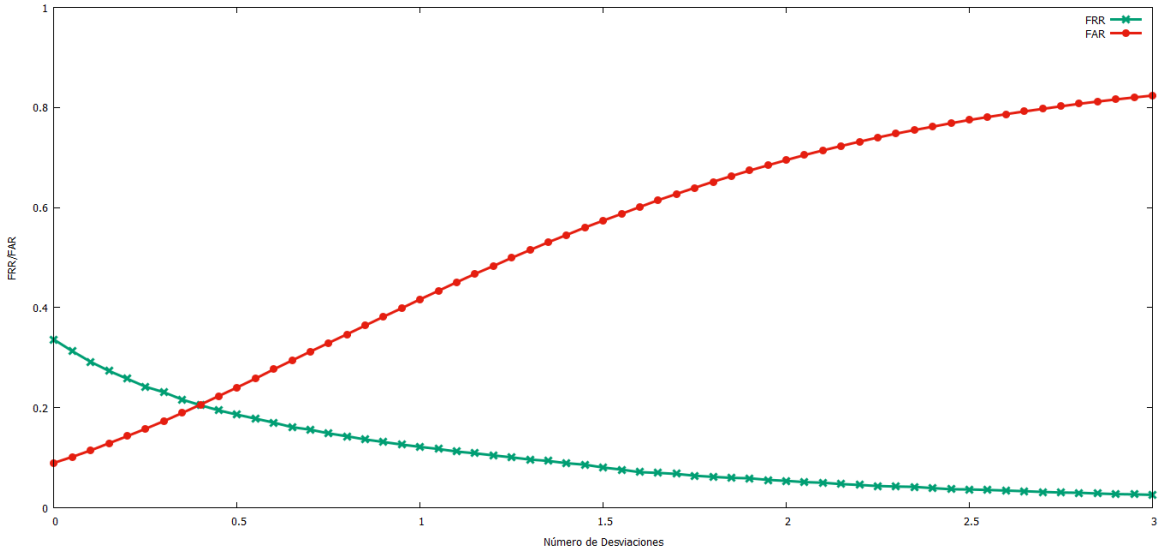



Figura 39: DT, FT1

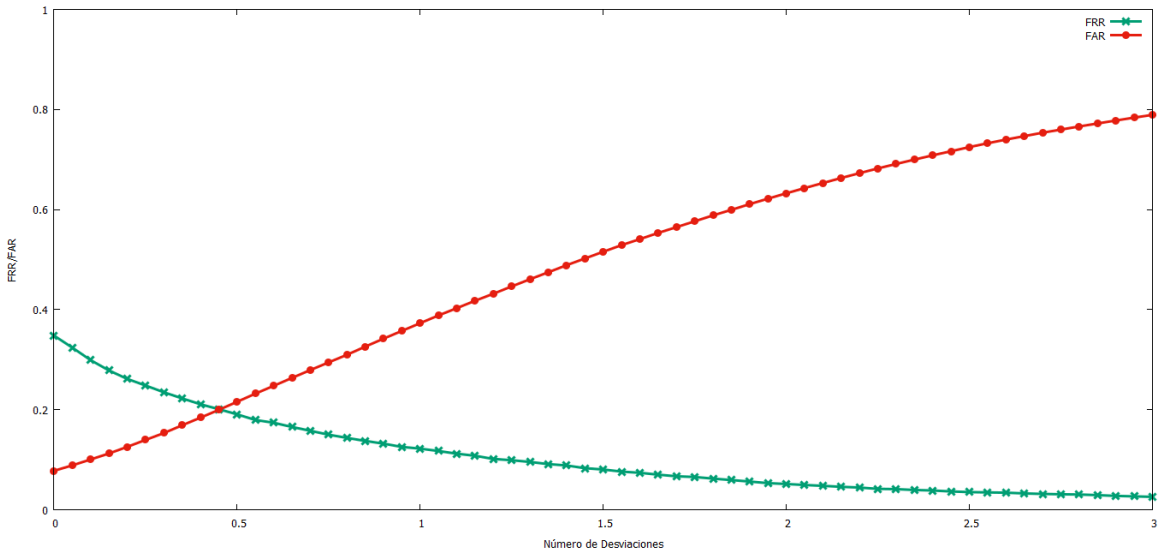


Figura 40: DT, FT2, FT3, FT4

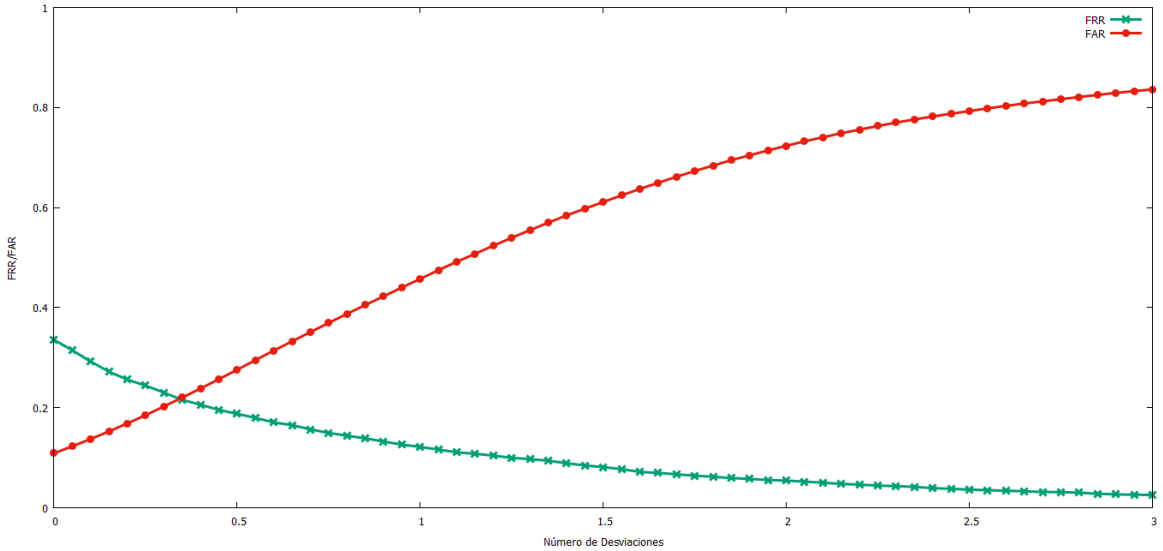



Figura 41: DT, FT2, FT3

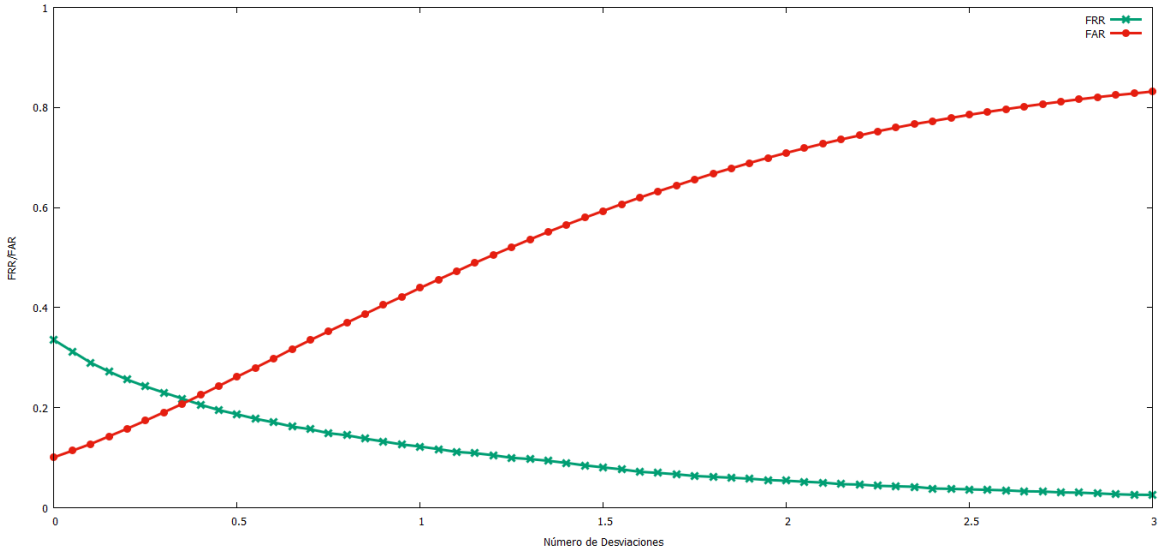



Figura 42: DT, FT2, FT4

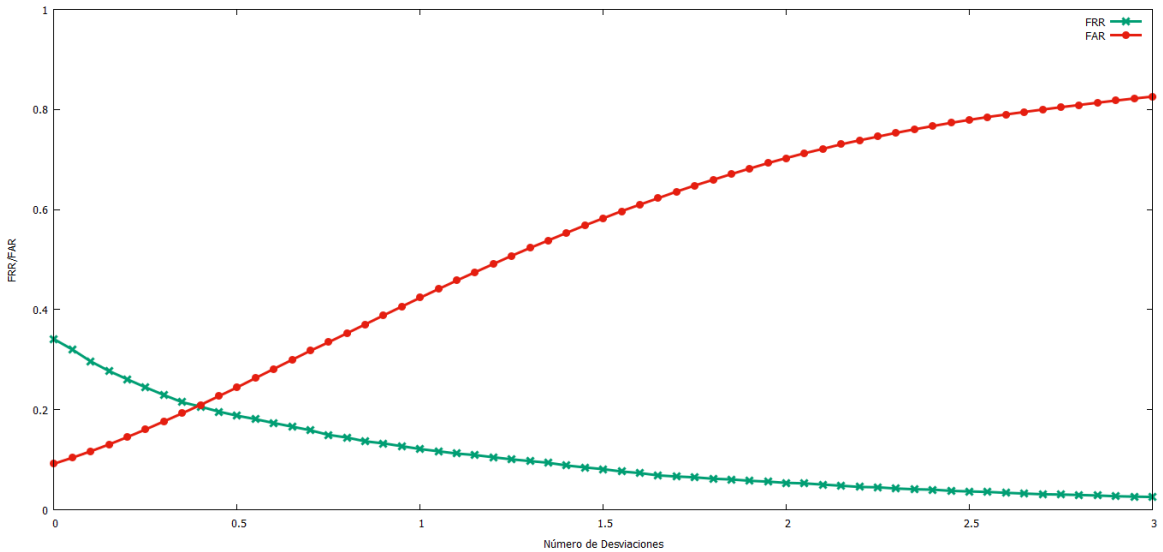


Figura 43: DT, FT2

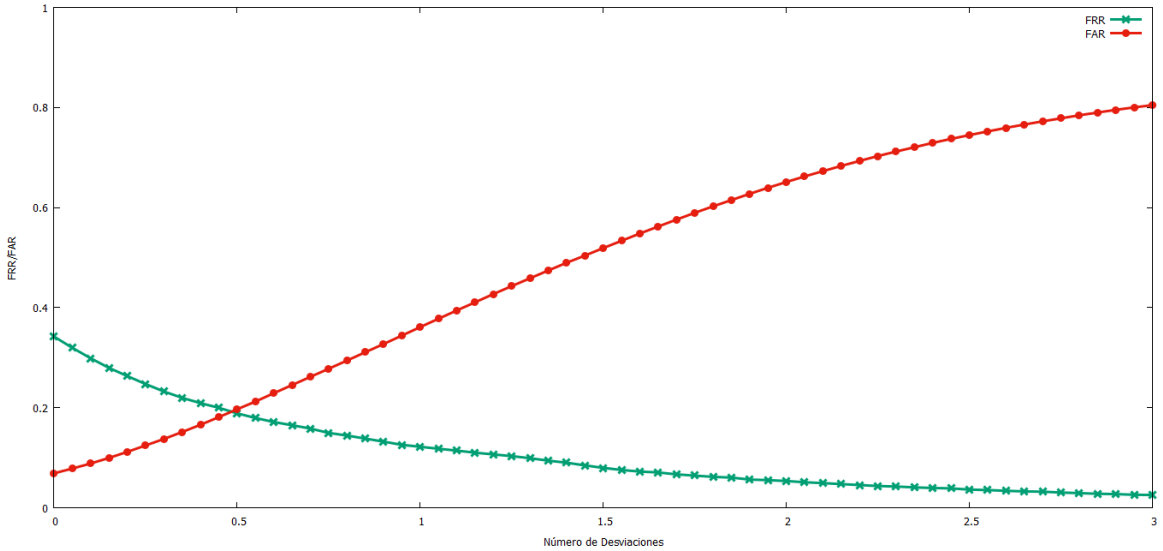



Figura 44: DT, FT3, FT4

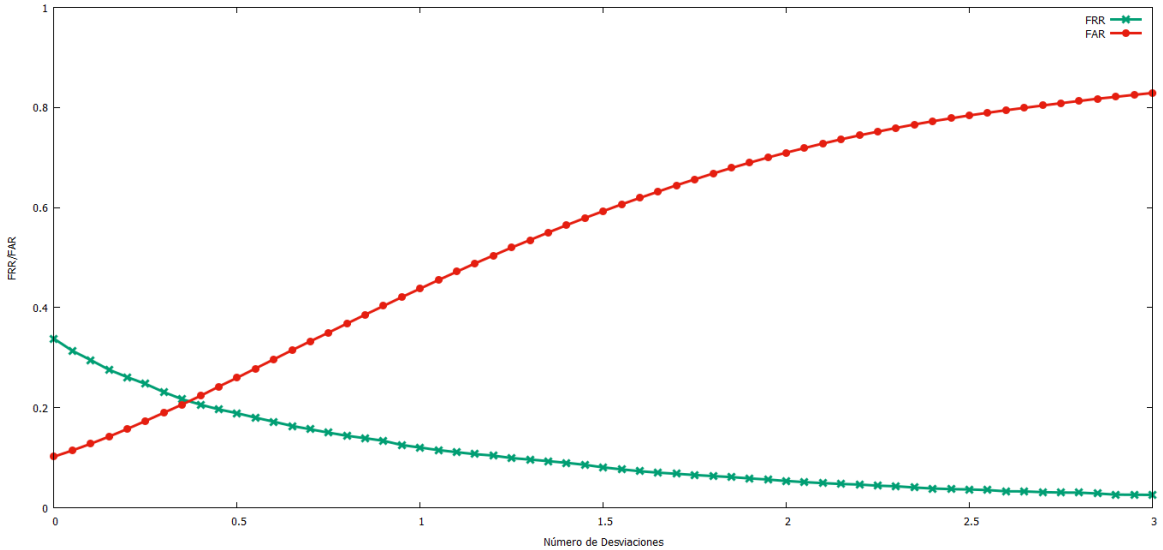



Figura 45: DT, FT3

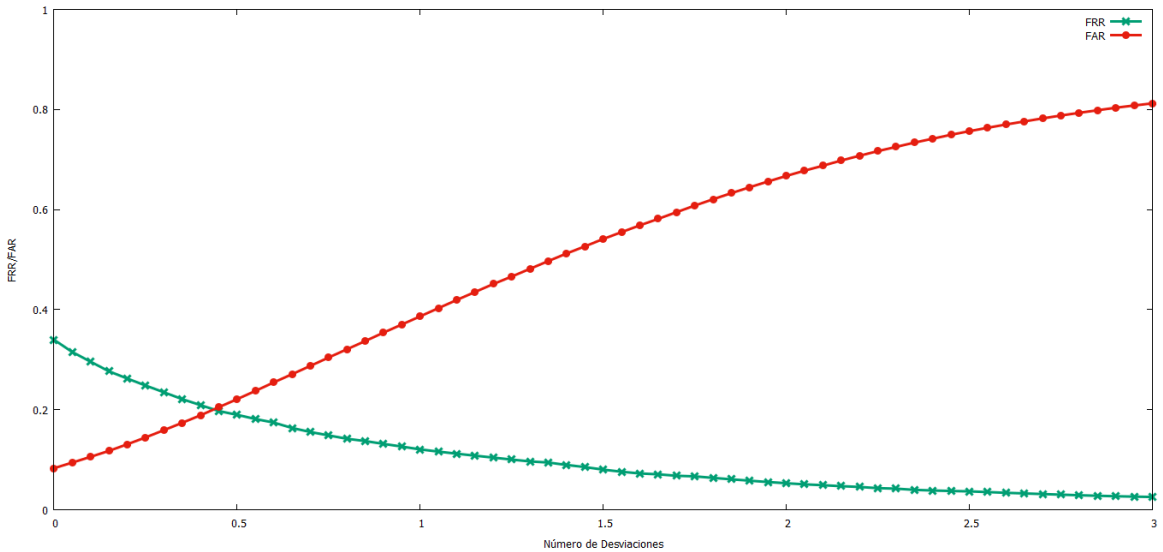


Figura 46: DT, FT4

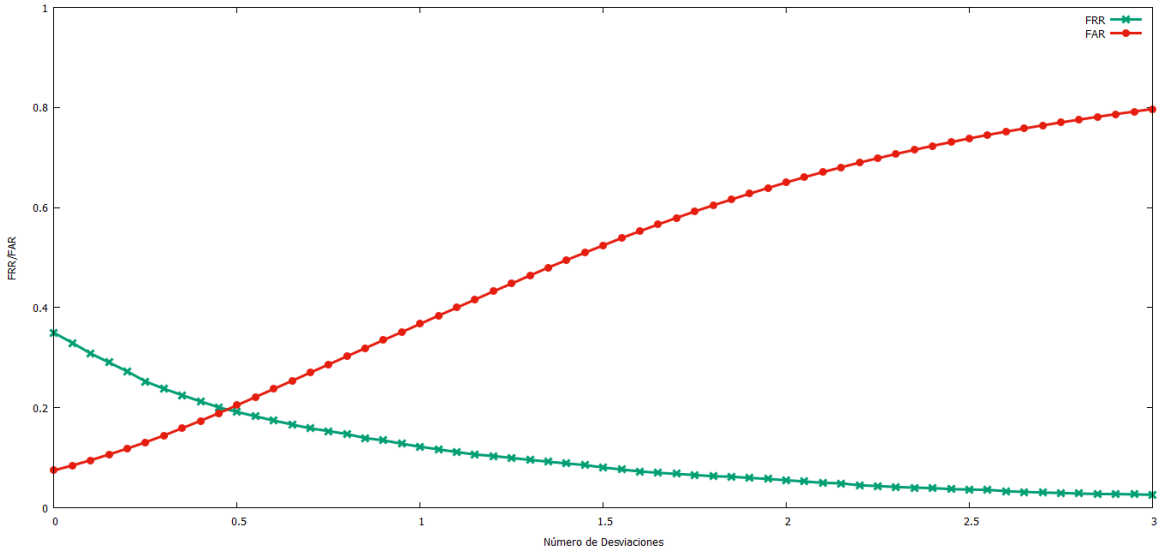



Figura 47: DT

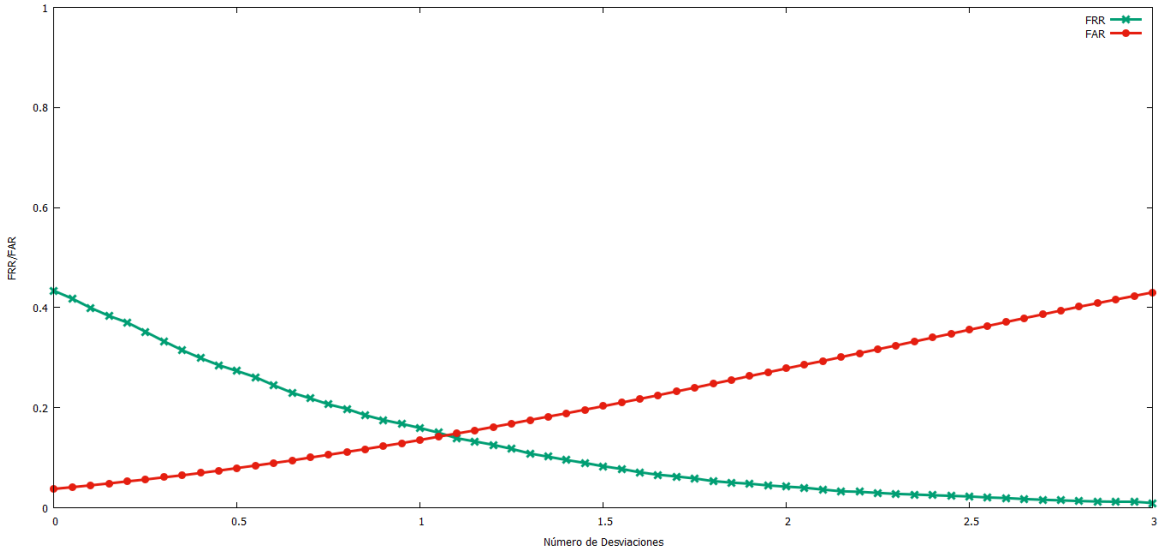



Figura 48: FT1, FT2, FT3, FT4

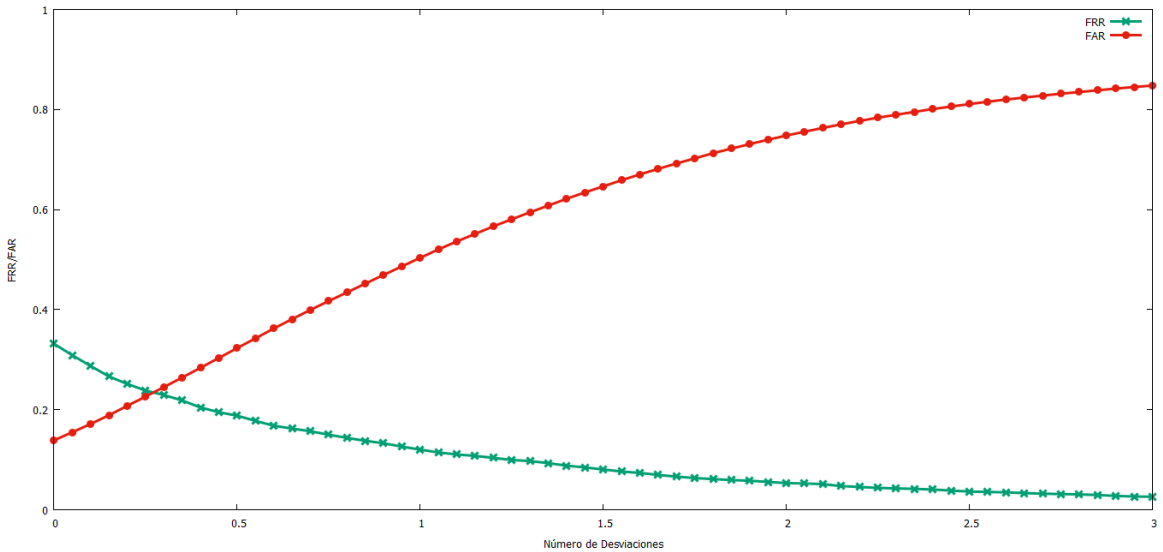


Figura 49: FT1, FT2, FT3

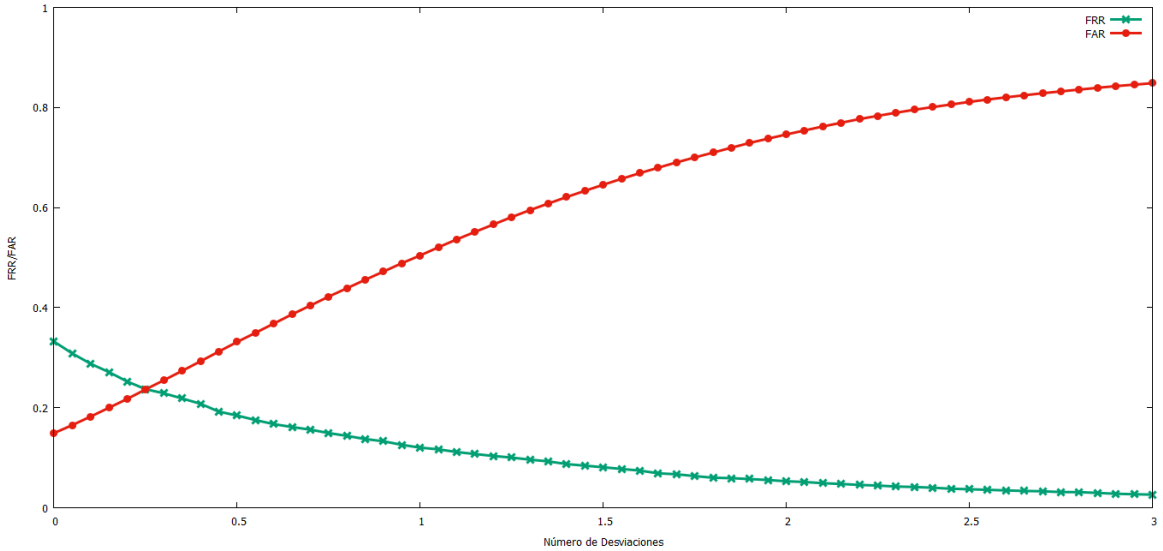



Figura 50: FT1, FT2, FT4

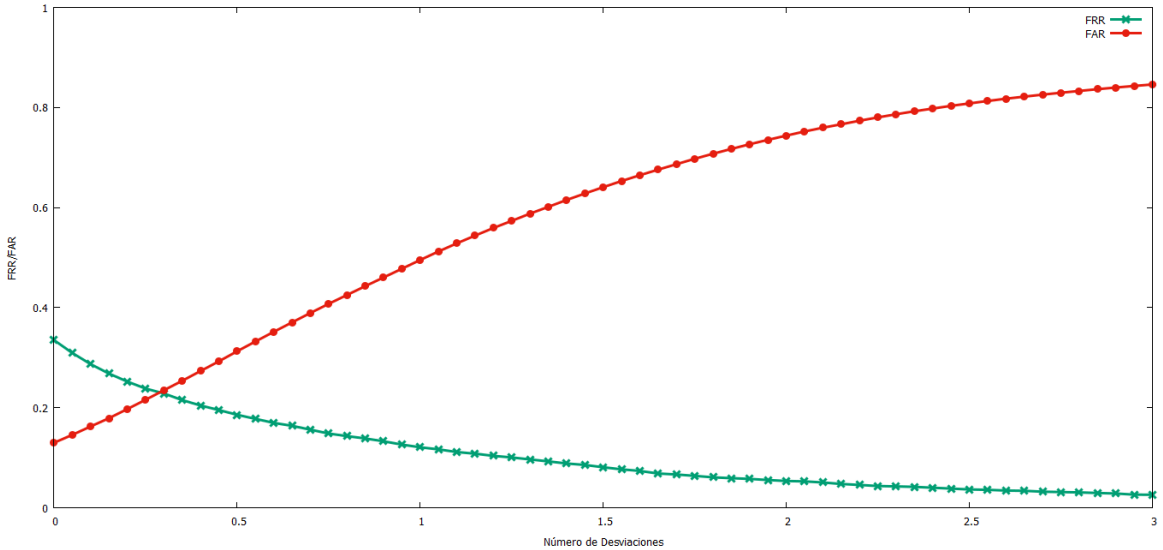



Figura 51: FT1, FT2

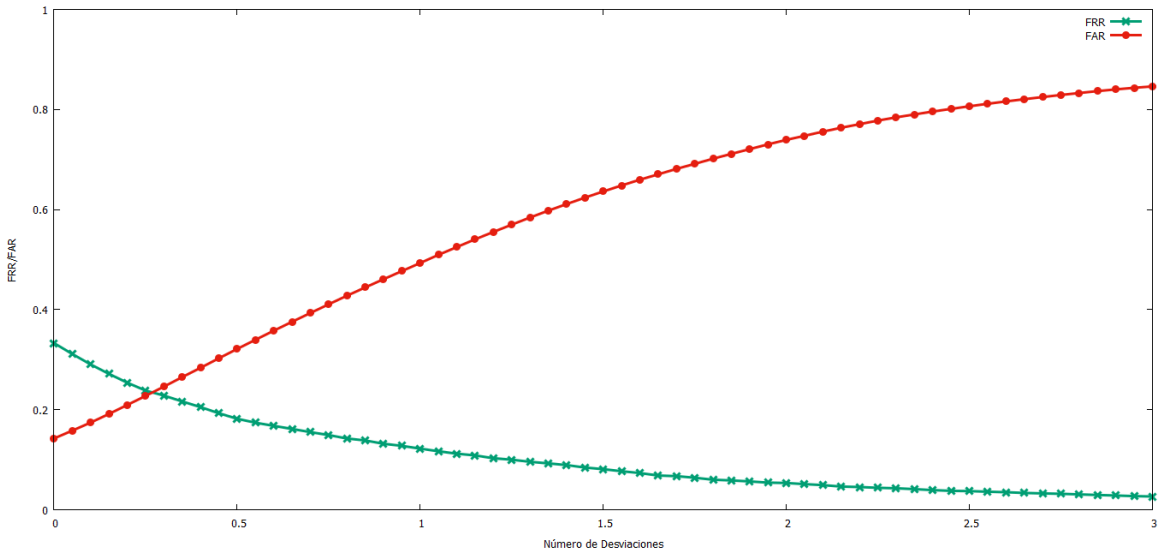


Figura 52: FT1, FT3, FT4

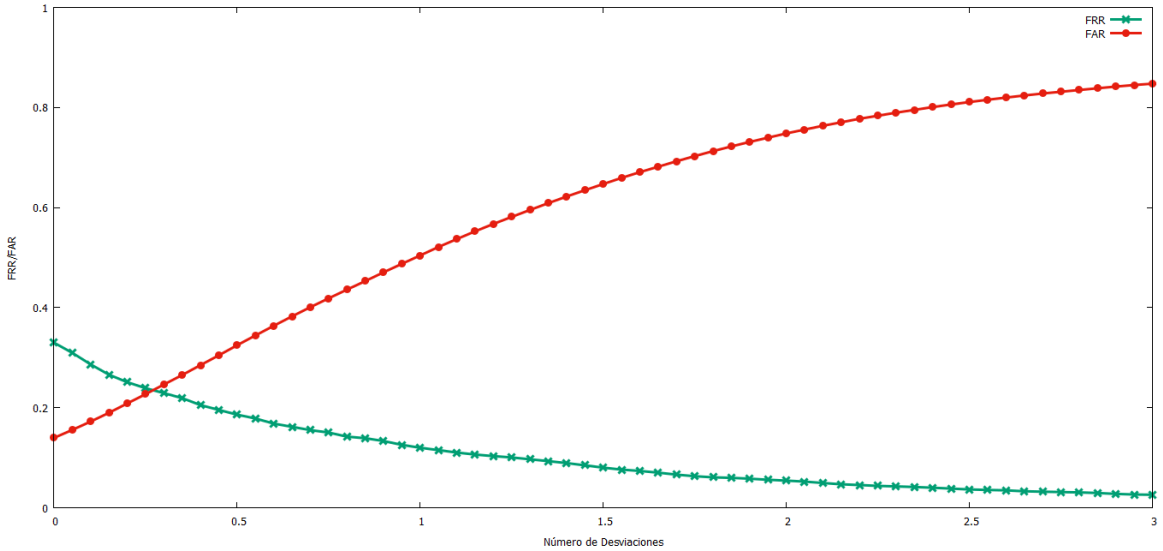



Figura 53: FT1, FT3

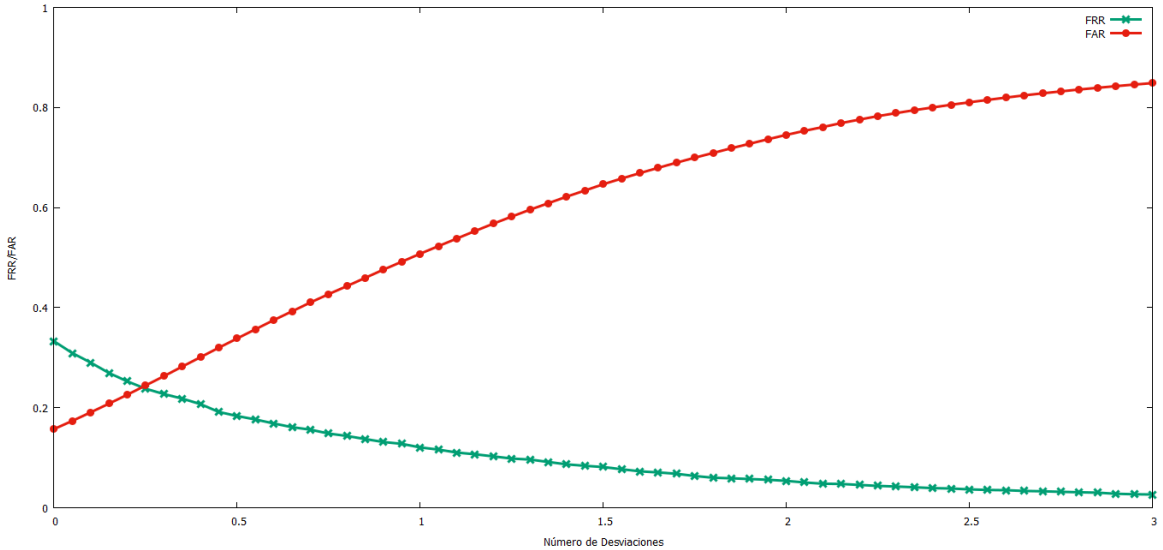



Figura 54: FT1, FT4

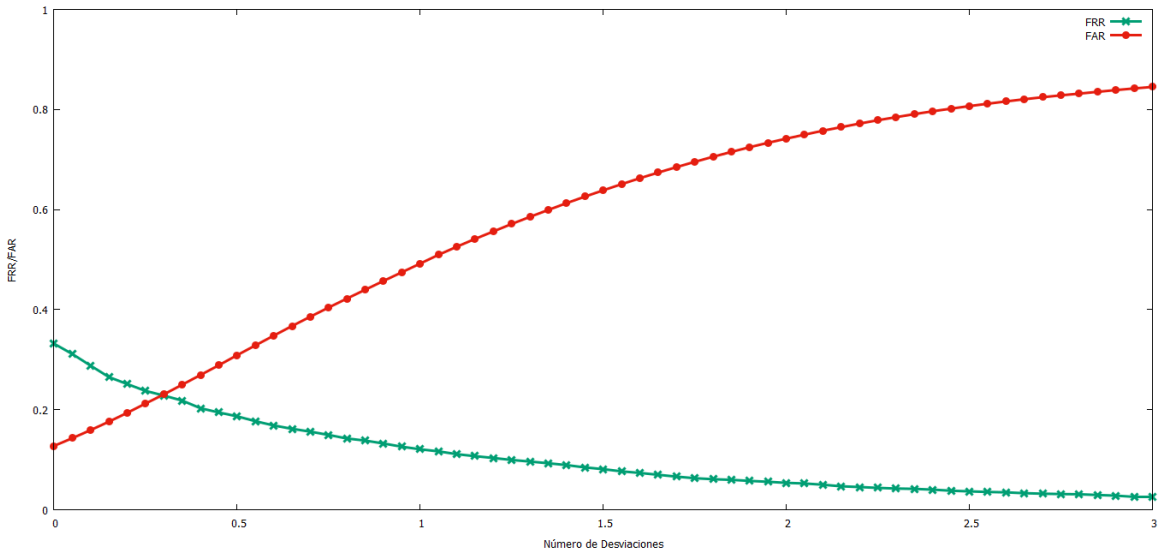


Figura 55: FT1

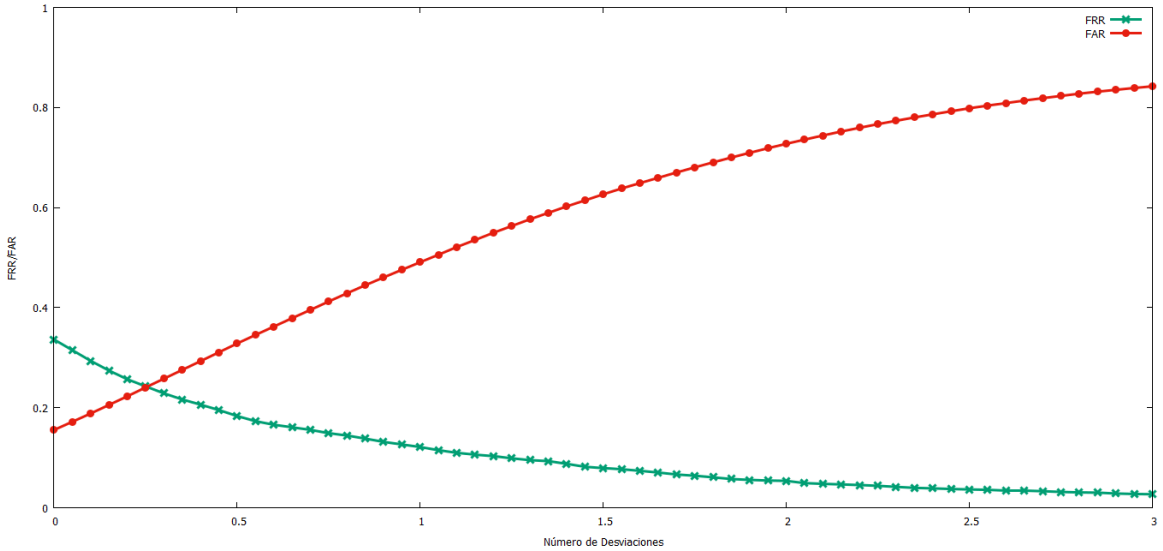



Figura 56: FT2, FT3, FT4

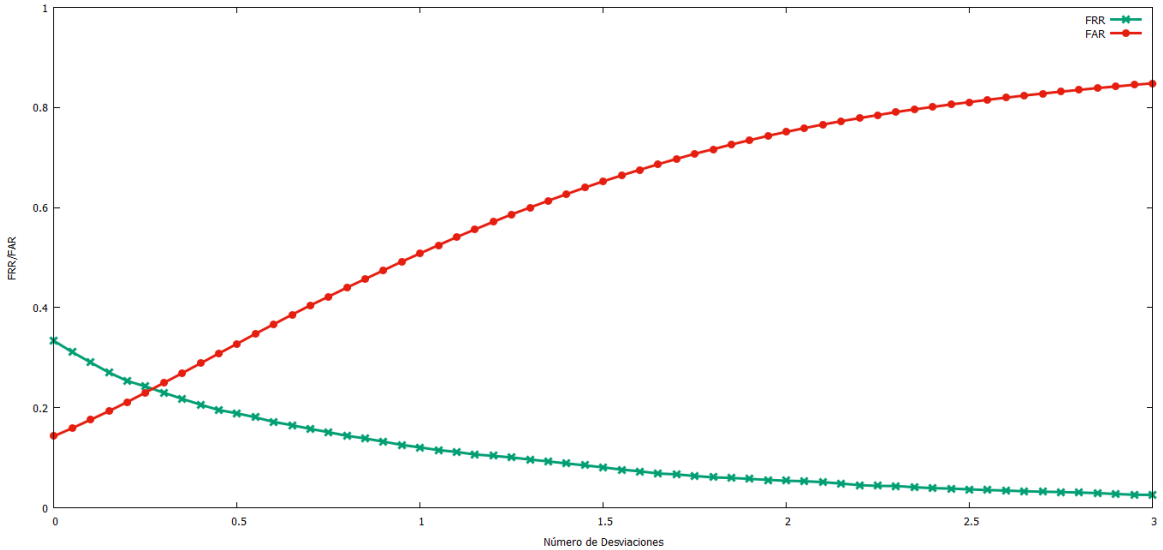



Figura 57: FT2, FT3

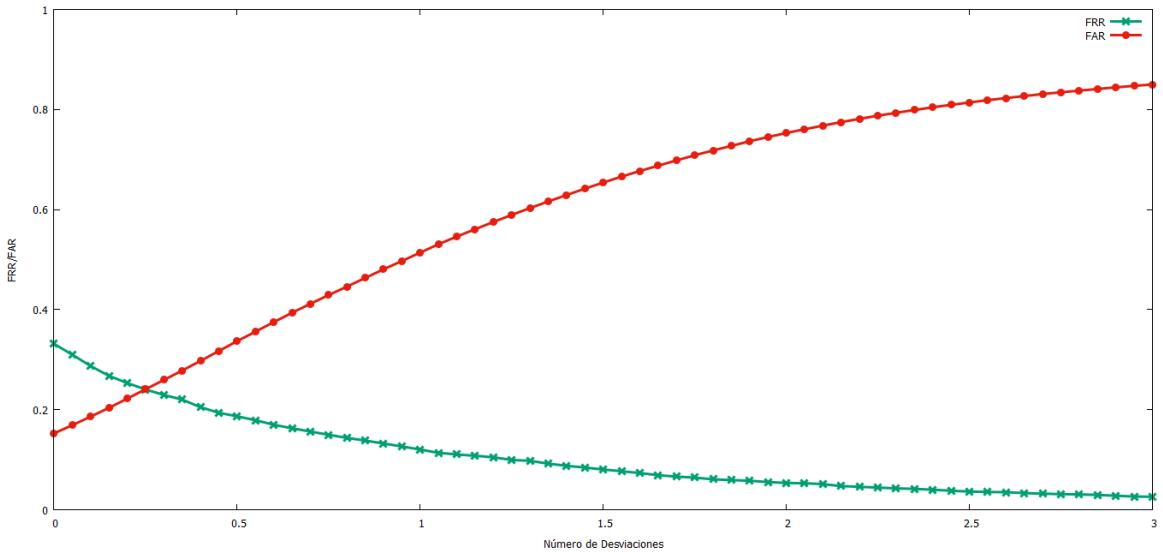


Figura 58: FT2, FT4

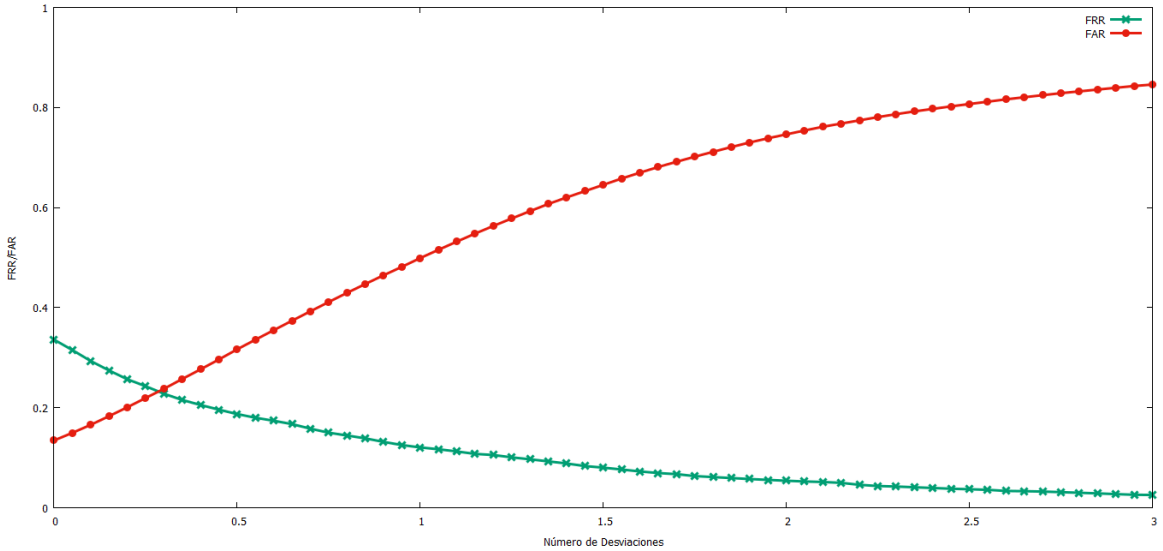



Figura 59: FT2

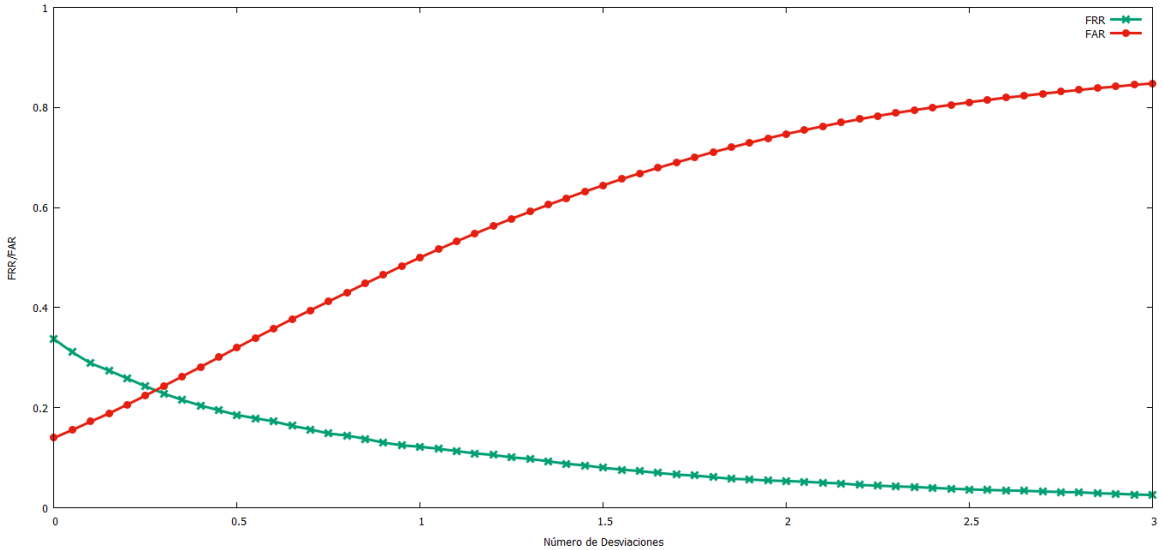



Figura 60: FT3, FT4

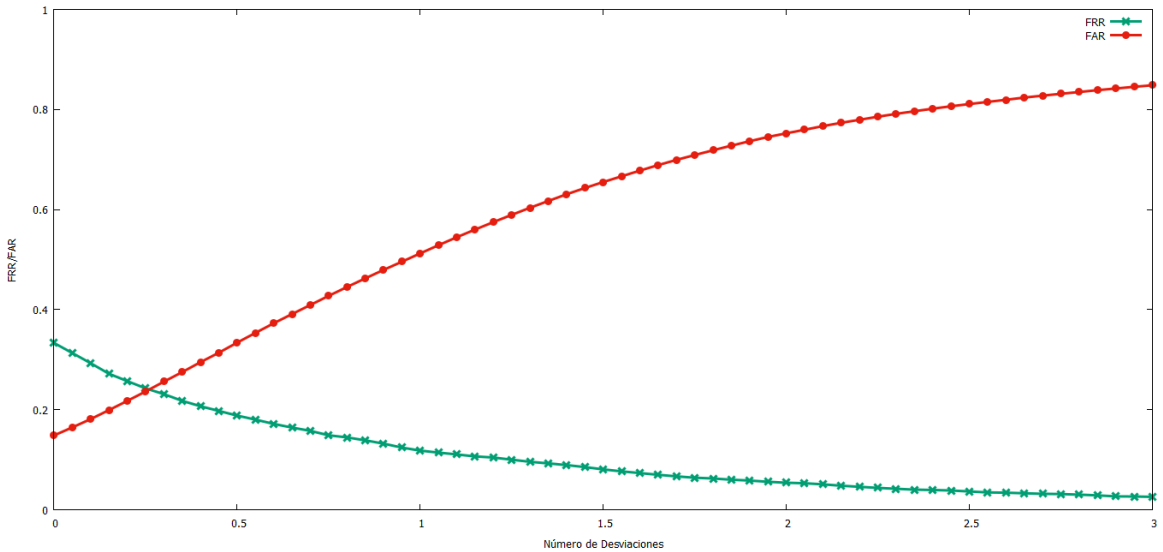


Figura 61: FT3

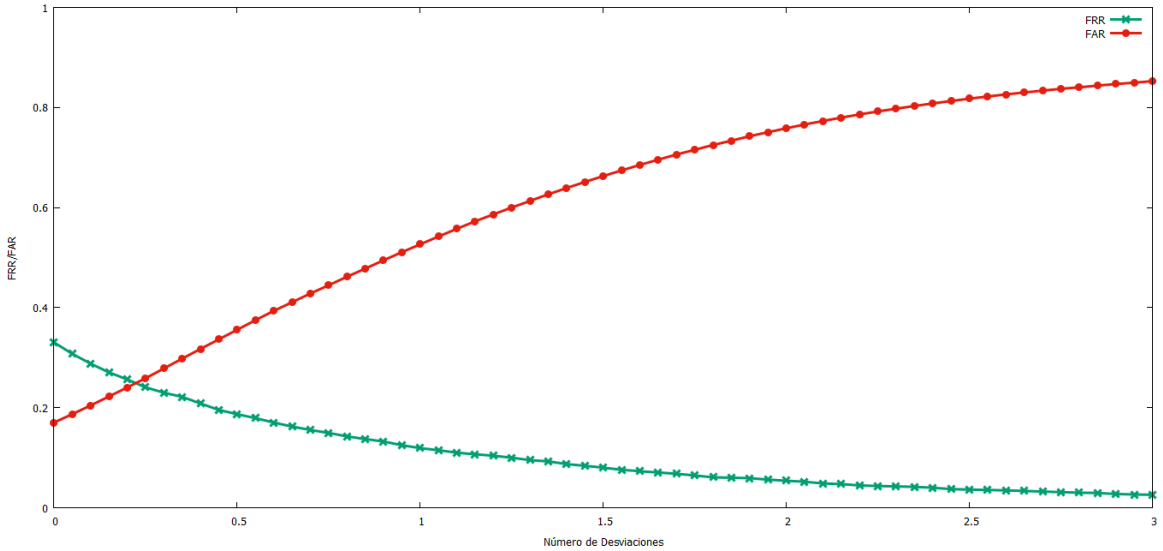
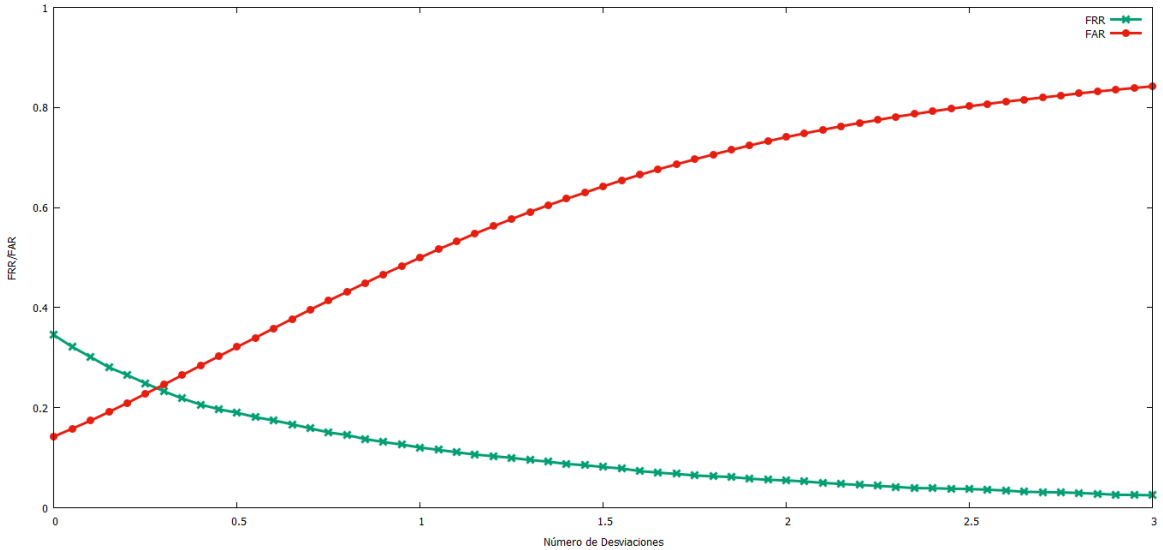



Figura 62: FT4



# Anexo III: Gráficas ZSCORE

**Figura 63: DT, FT1, FT2, FT3, FT4**

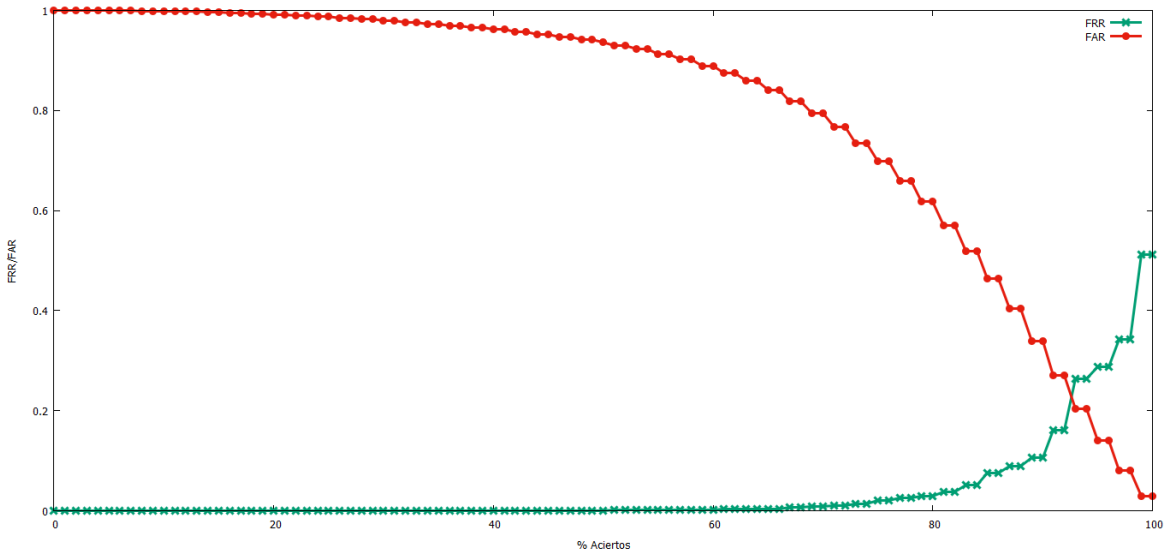



**Figura 64: DT, FT1, FT2, FT3**

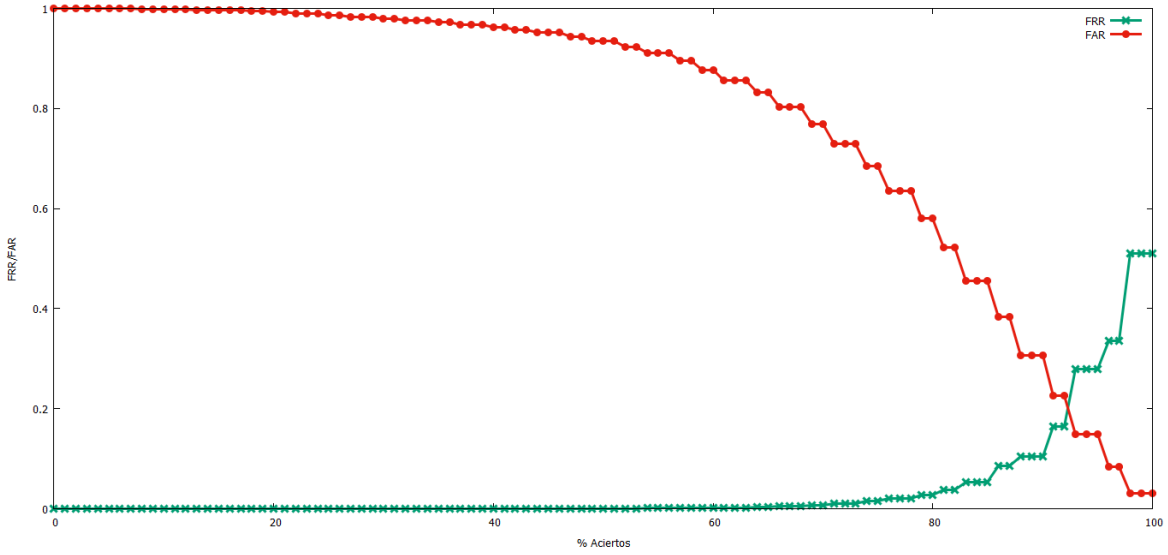


Figura 65: DT, FT1, FT2, FT4

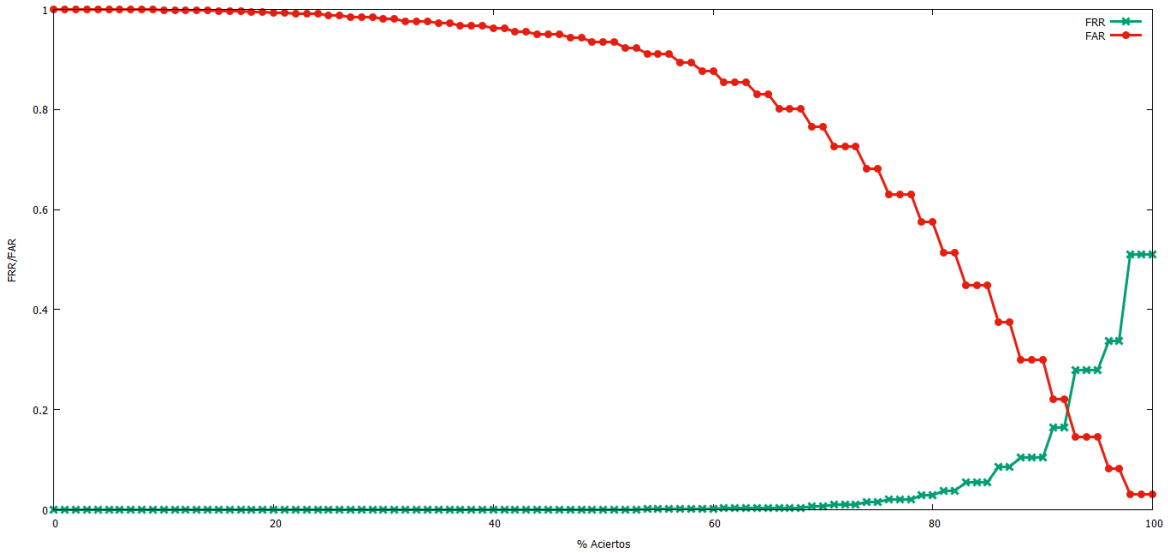



Figura 66: DT, FT1, FT2

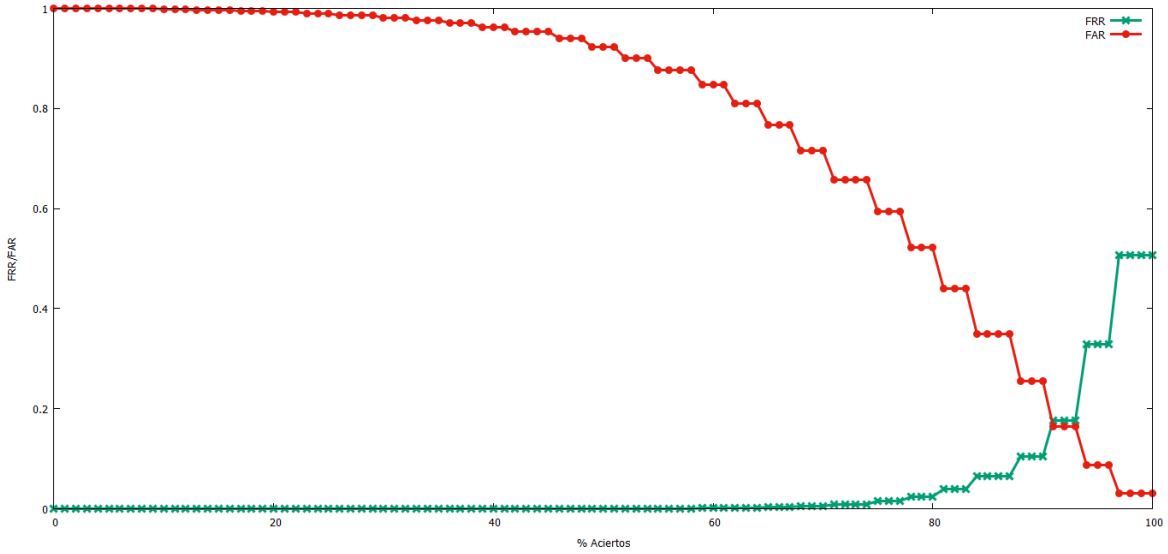



Figura 67: DT, FT1, FT3, FT4

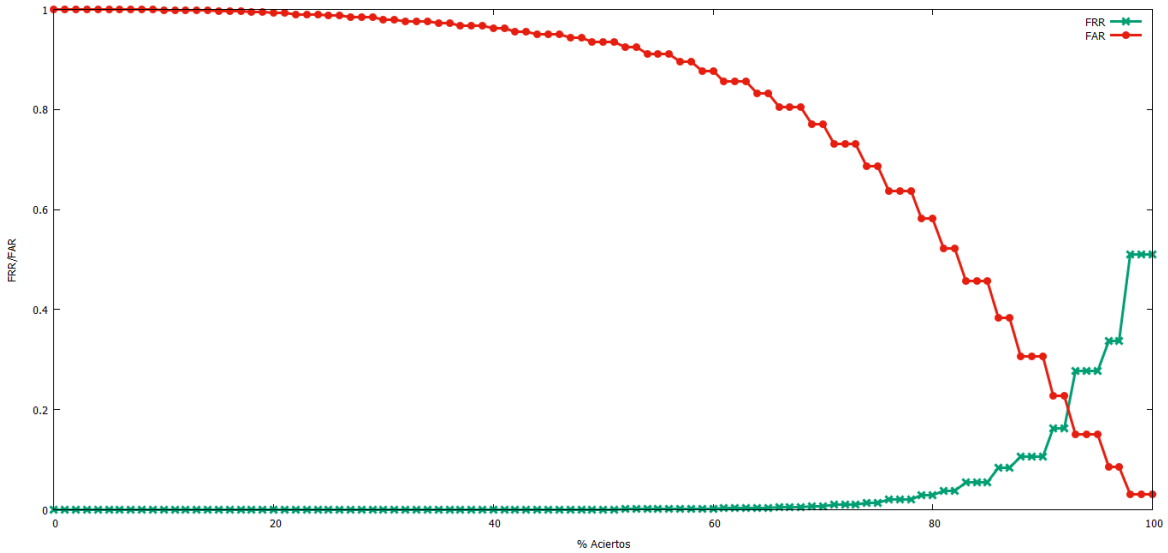


Figura 68: DT, FT1, FT3

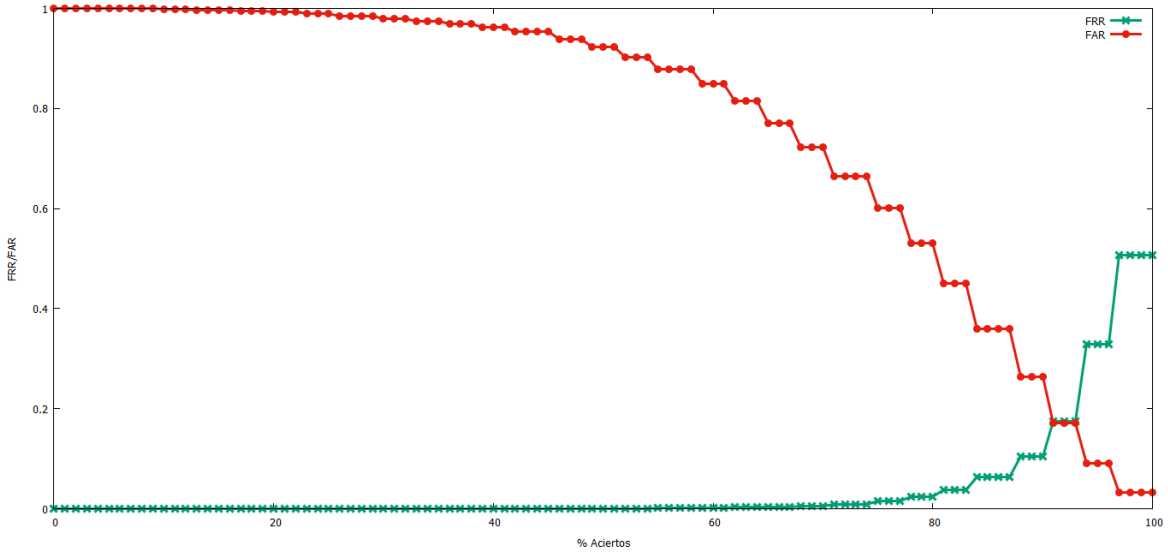



Figura 69: DT, FT1, FT4

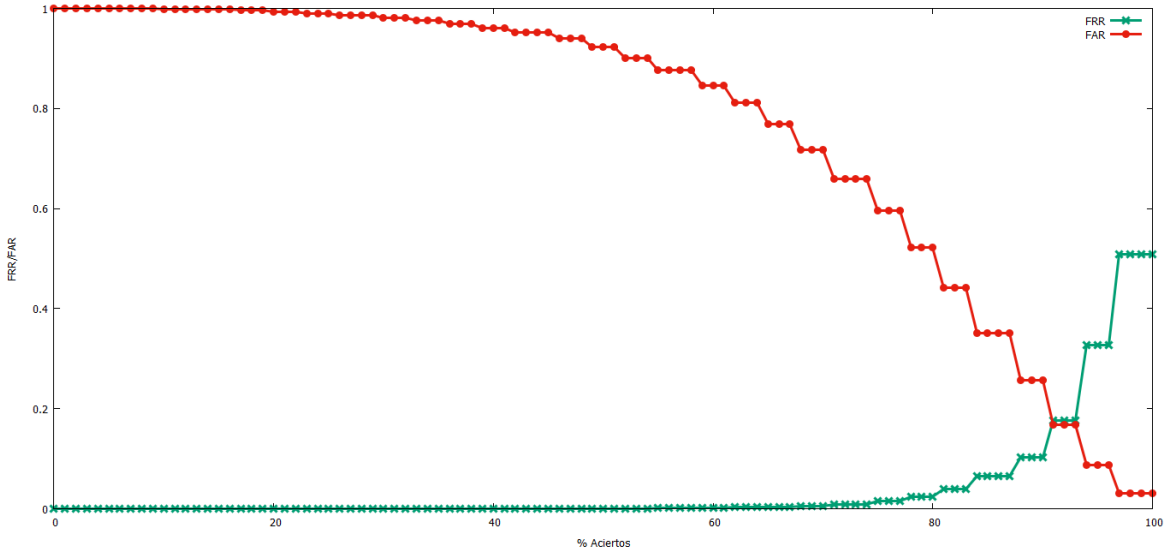



Figura 70: DT, FT1

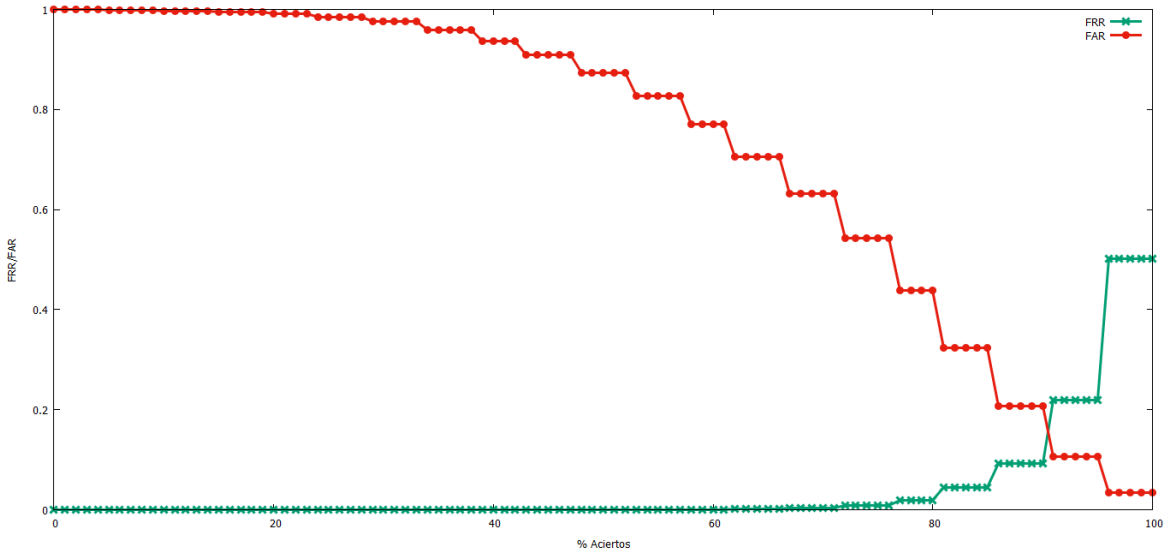


Figura 71: DT, FT2, FT3, FT4

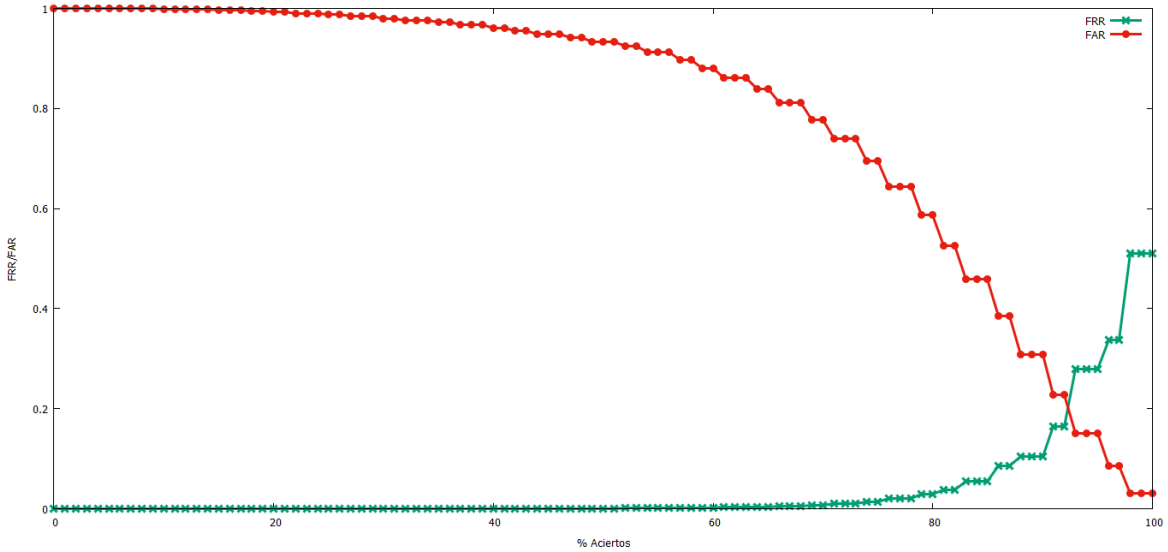



Figura 72: DT, FT2, FT3

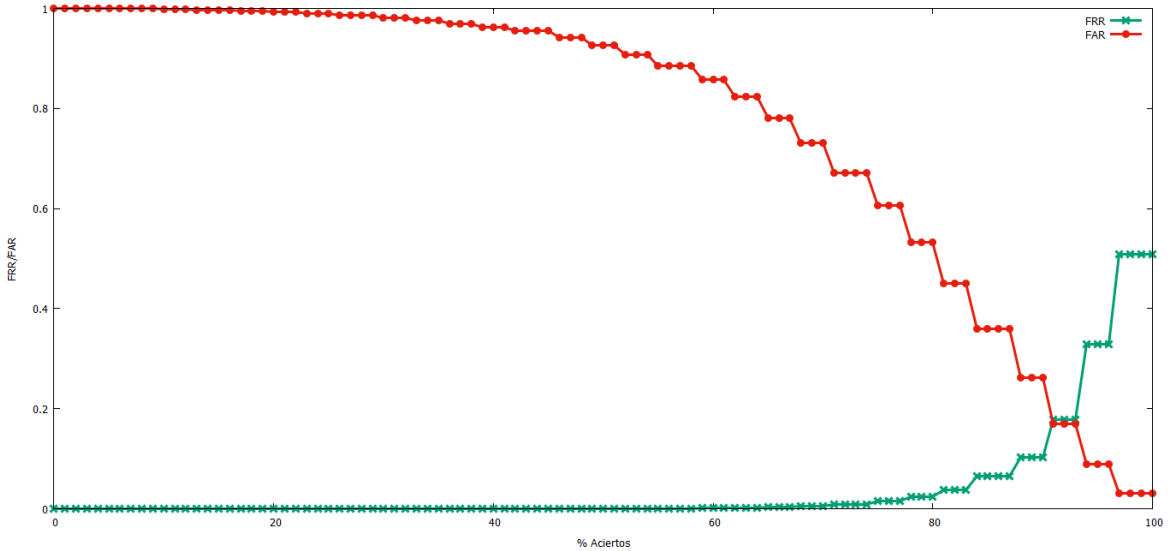



Figura 73: DT, FT2, FT4

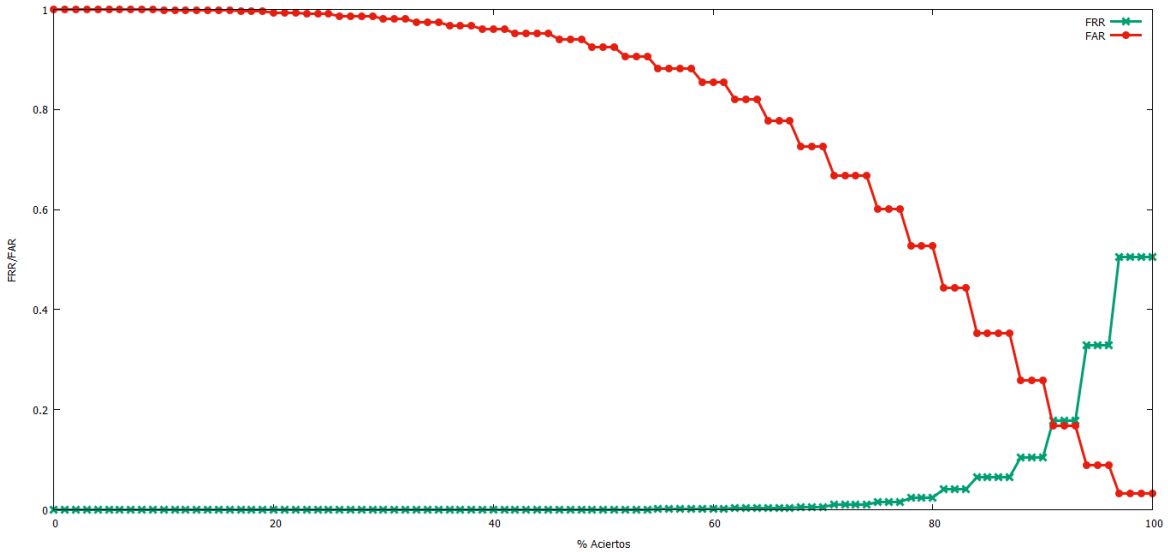


Figura 74: DT, FT2

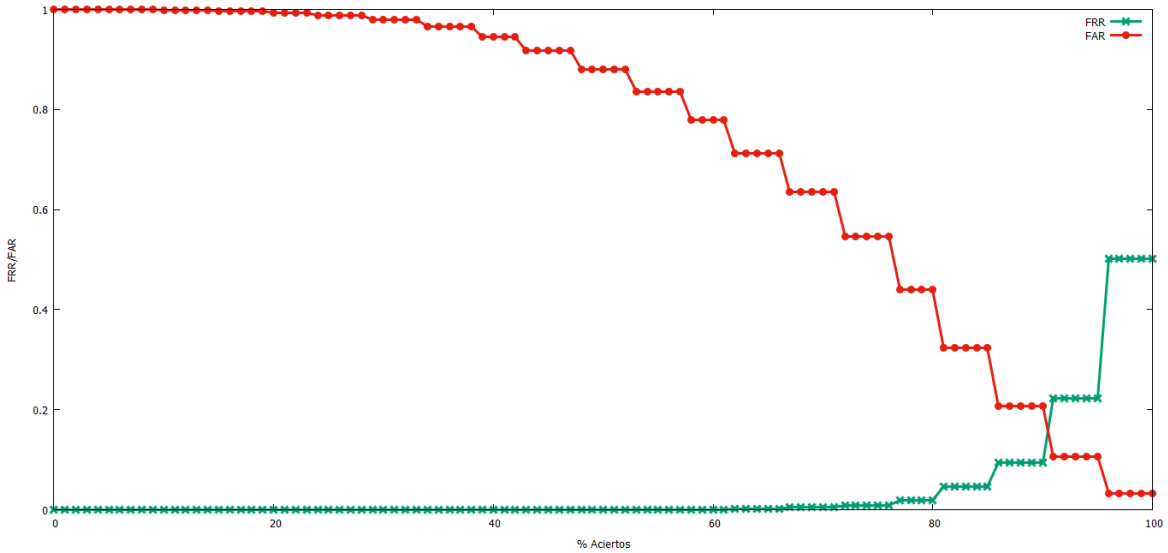



Figura 75: DT, FT3, FT4

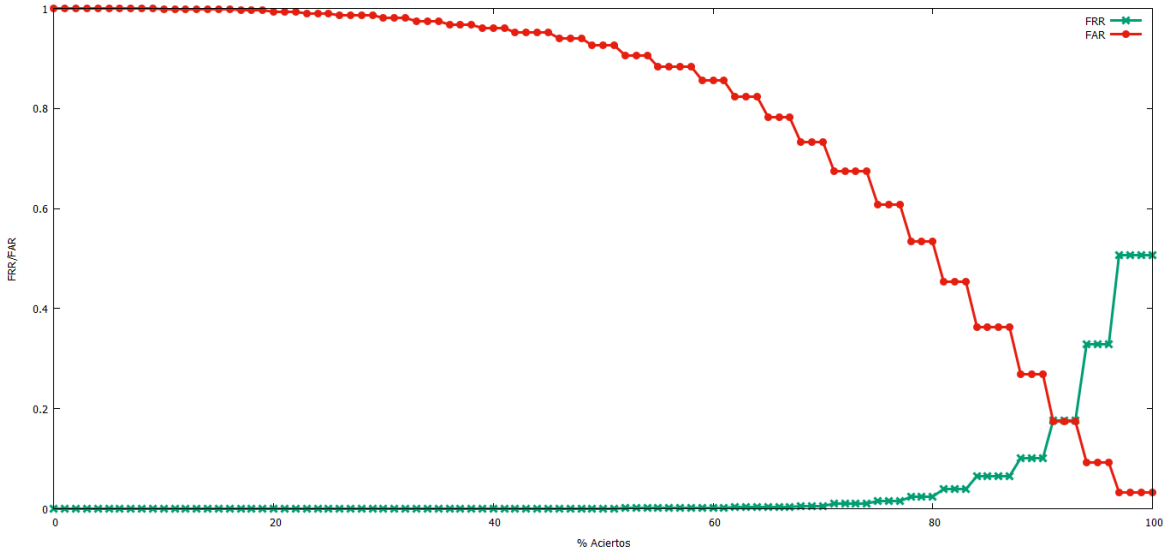



Figura 76: DT, FT3

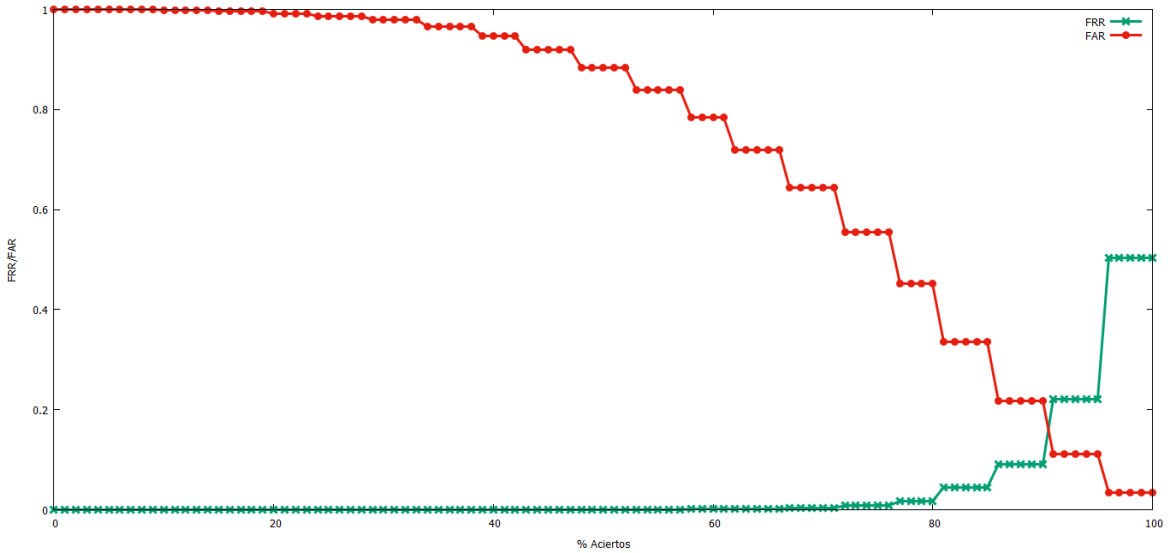


Figura 77: DT, FT4

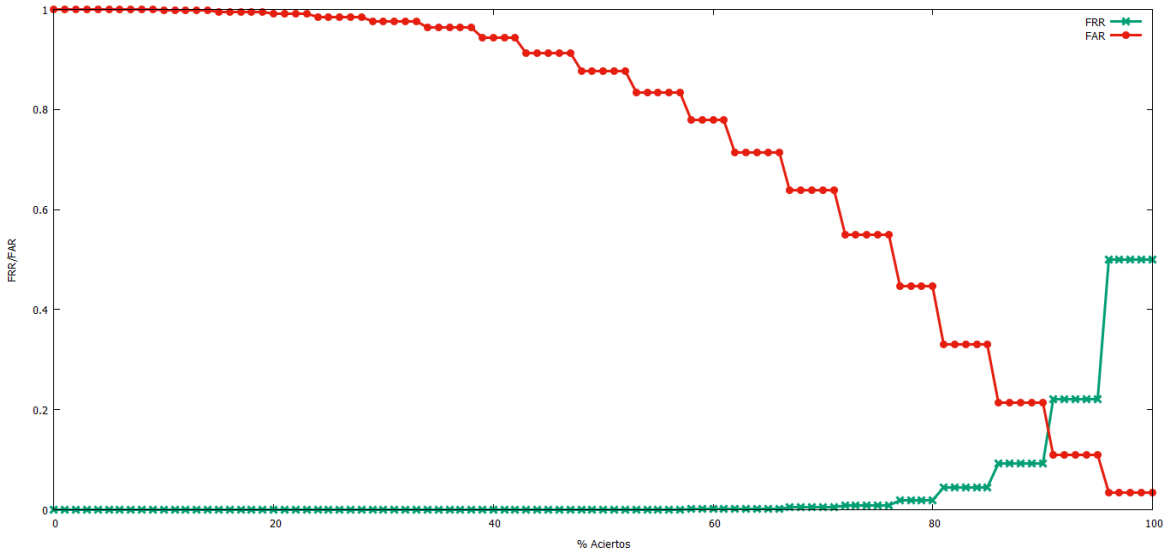



Figura 78: DT

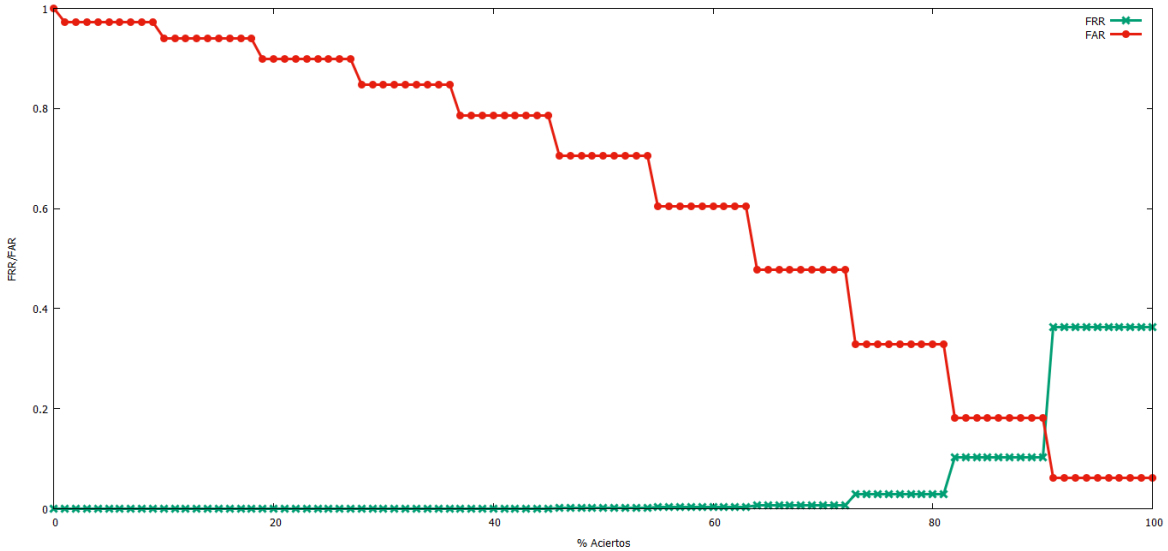



Figura 79: FT1, FT2, FT3, FT4

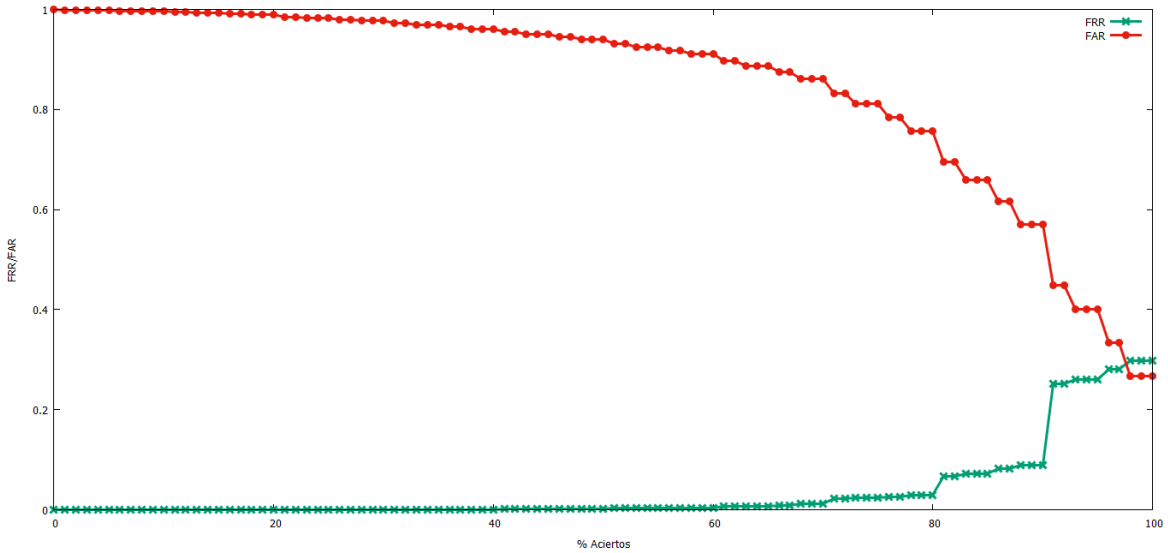


Figura 80: FT1, FT2, FT3

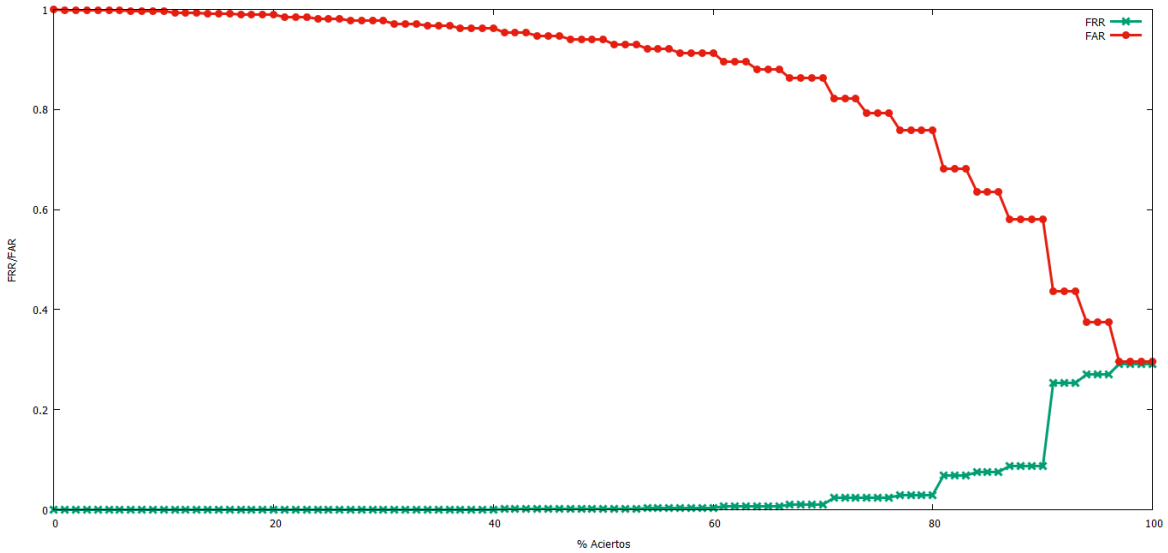



Figura 81: FT1, FT2, FT4

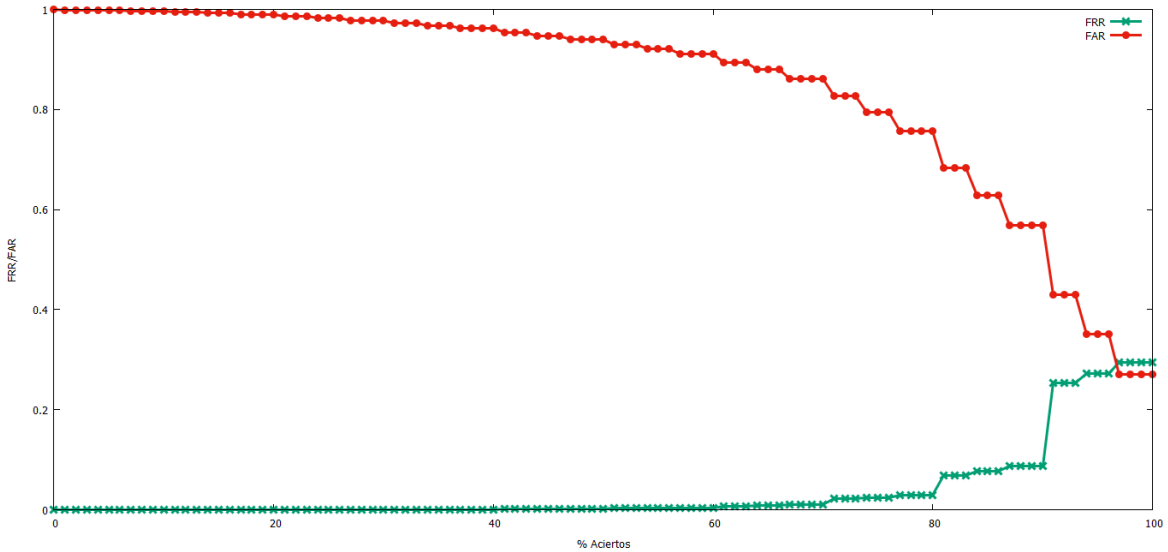



Figura 82: FT1, FT2

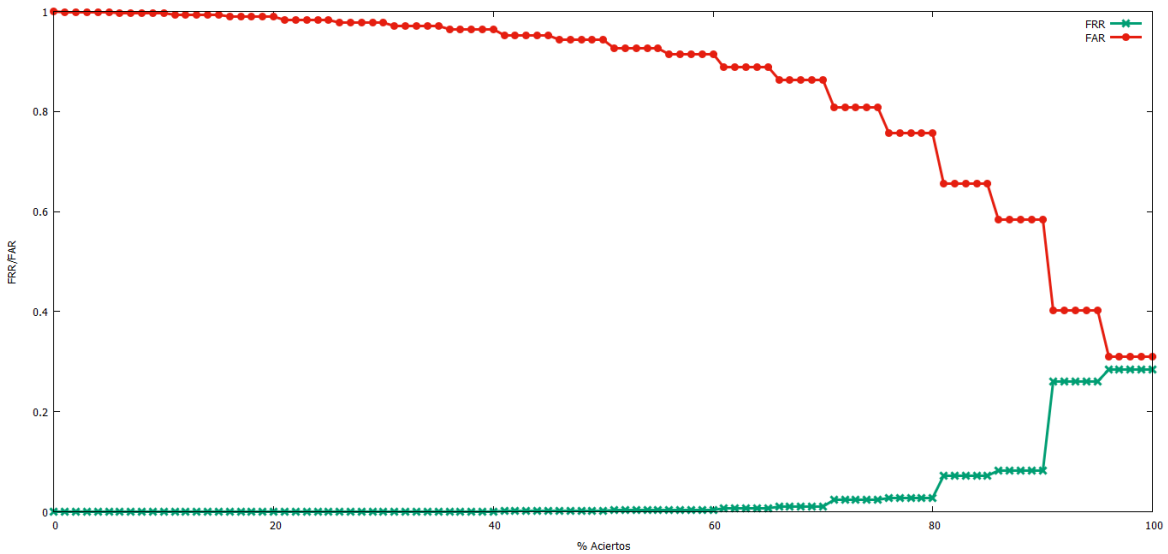


Figura 83: FT1, FT3, FT4

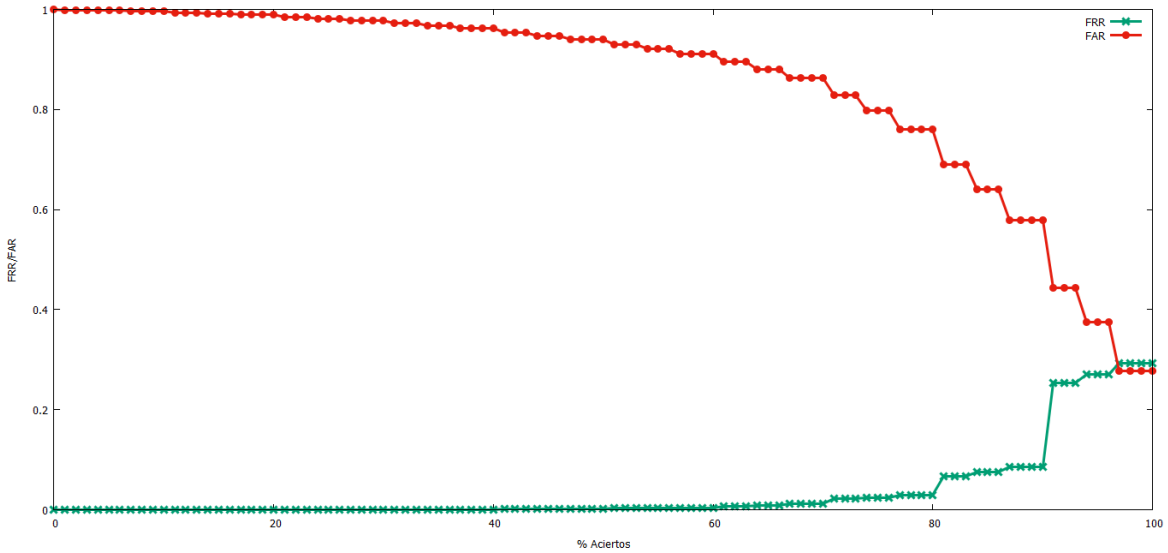



Figura 84: FT1, FT3

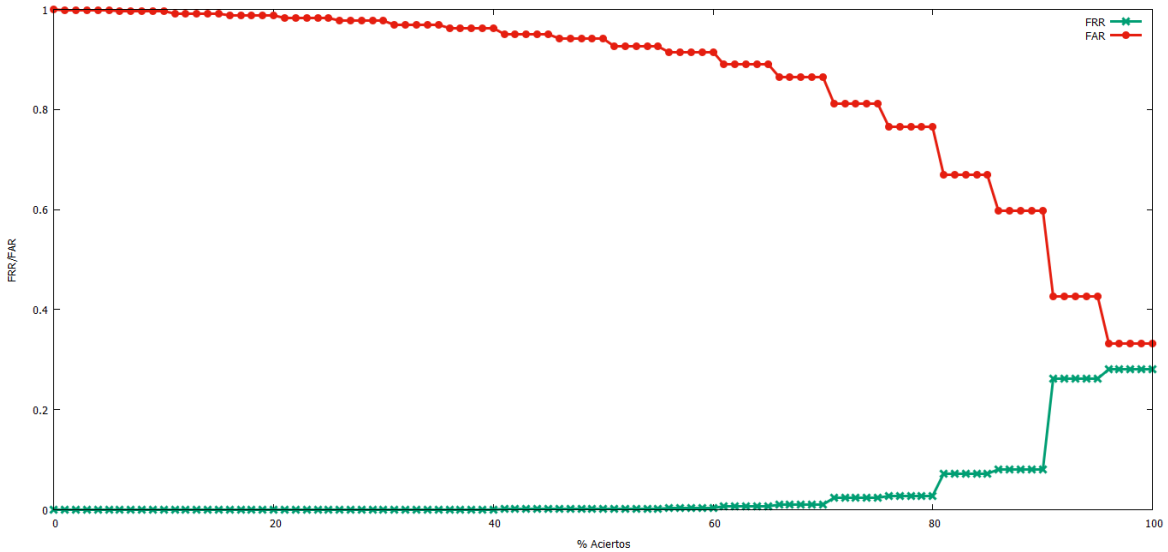



Figura 85: FT1, FT4

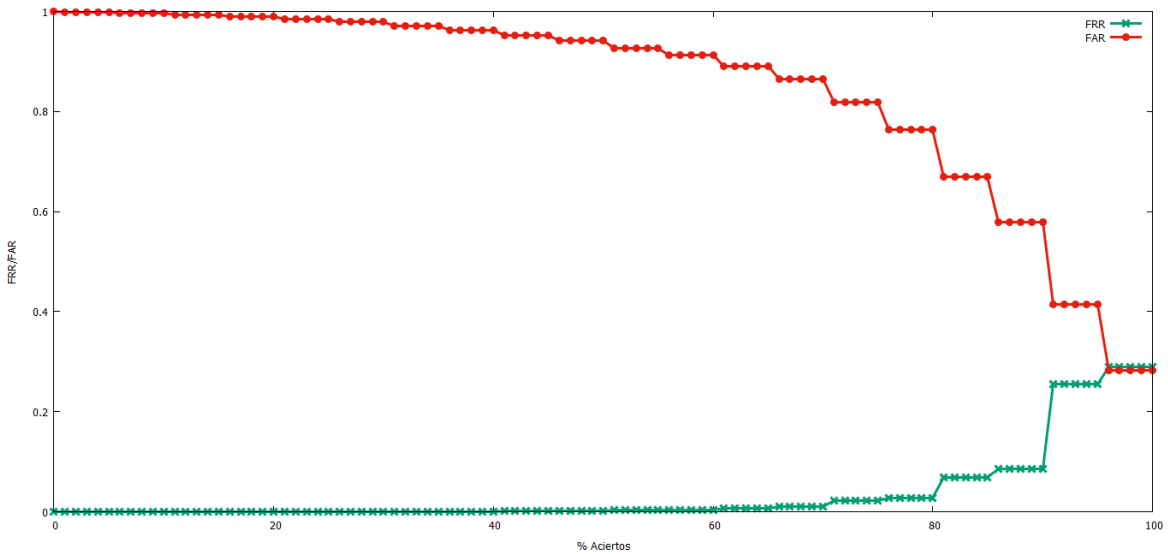


Figura 86: FT1

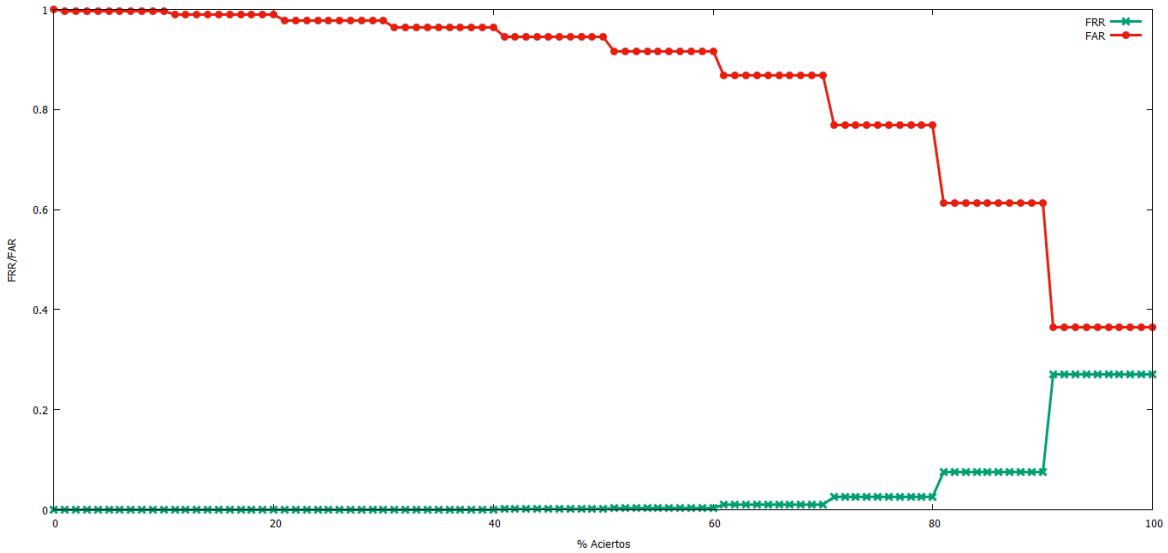



Figura 87: FT2, FT3, FT4

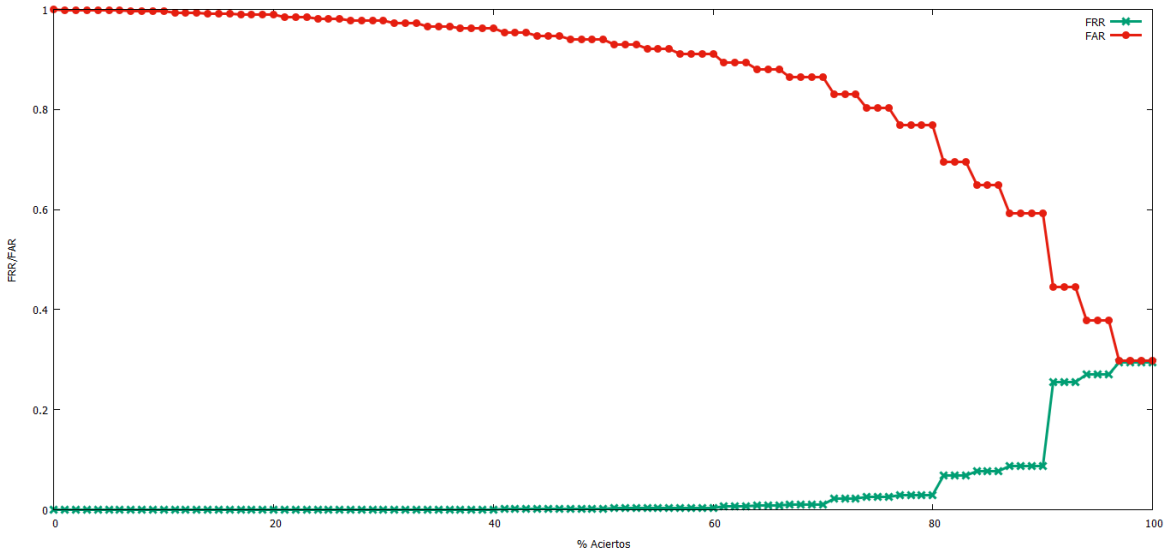



Figura 88: FT2, FT3

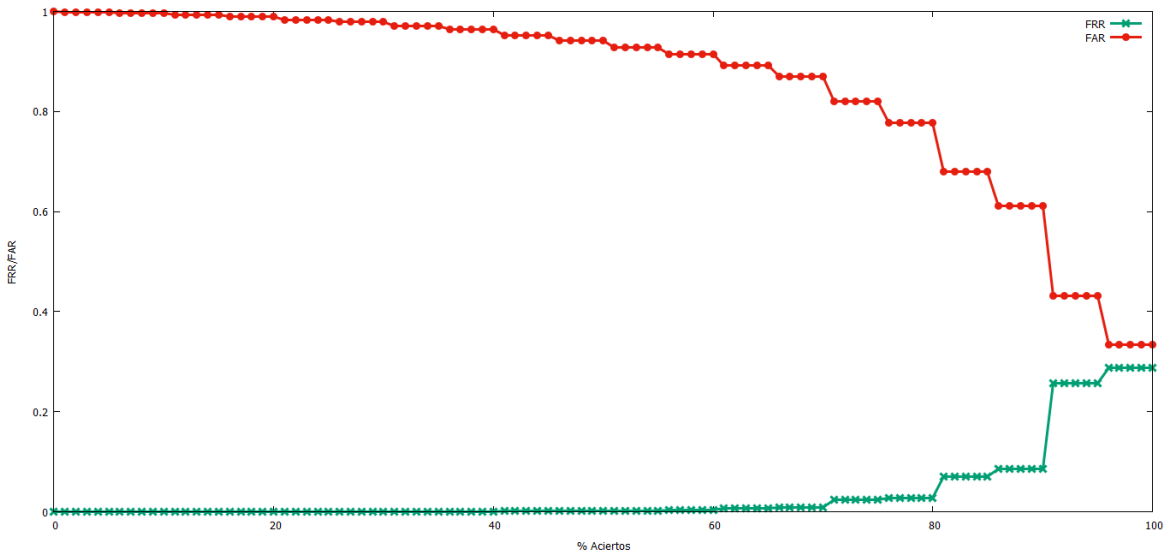


Figura 89: FT2, FT4

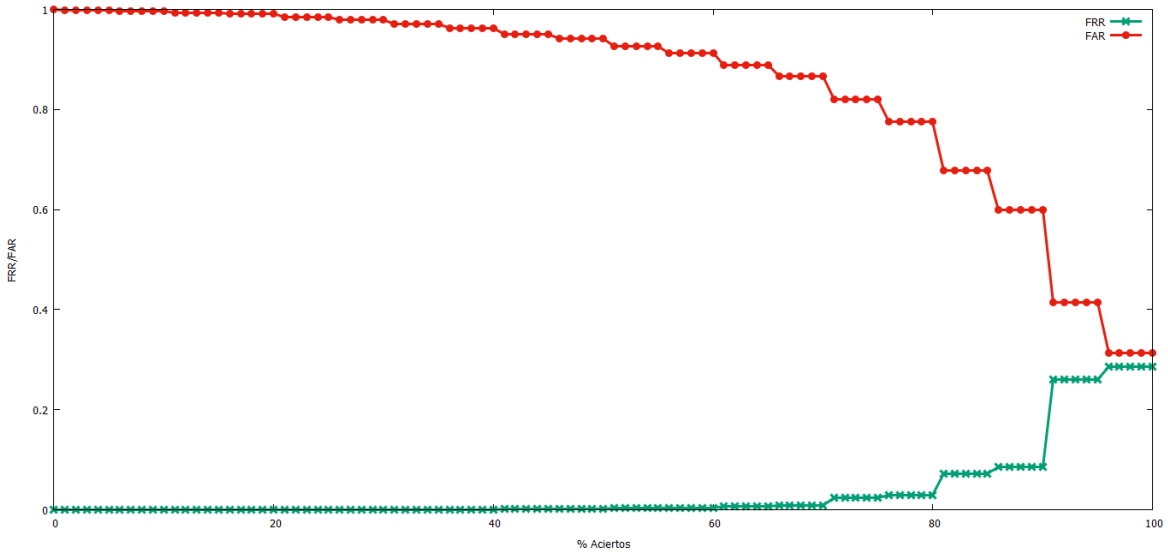



Figura 90: FT2

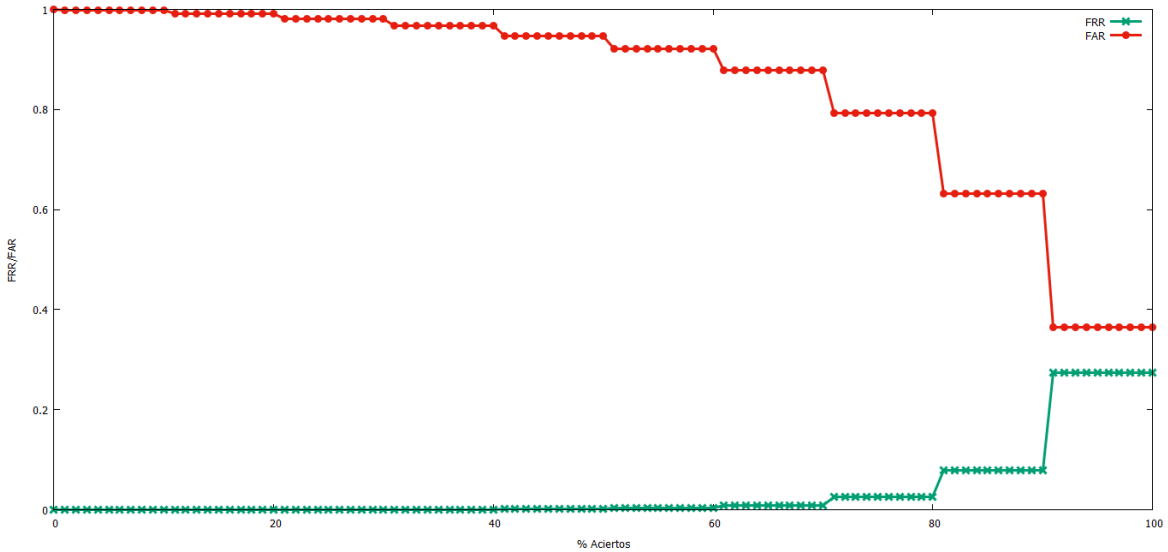



Figura 91: FT3, FT4

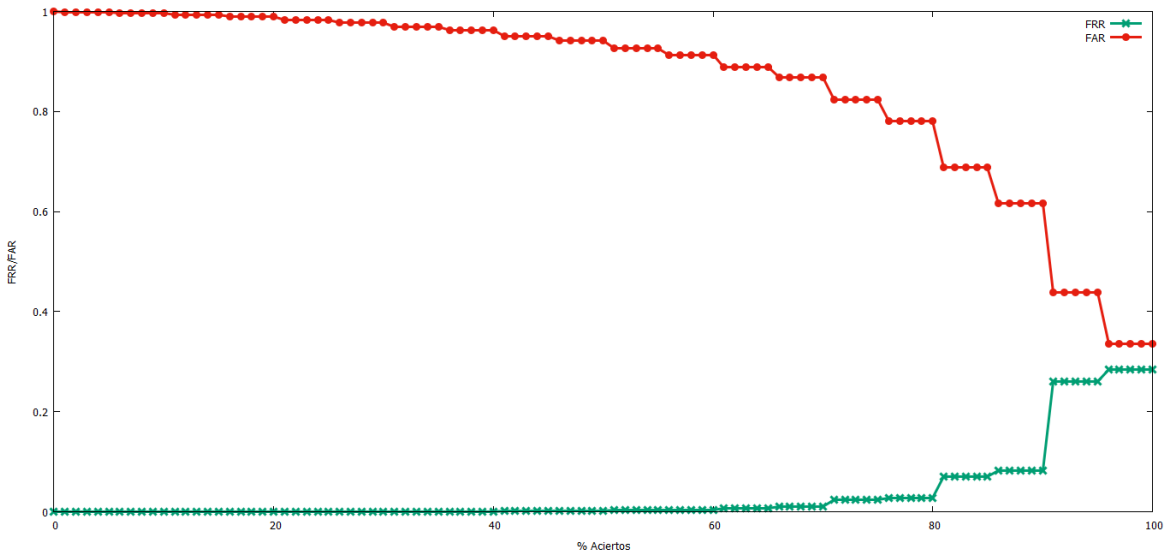


Figura 92: FT3

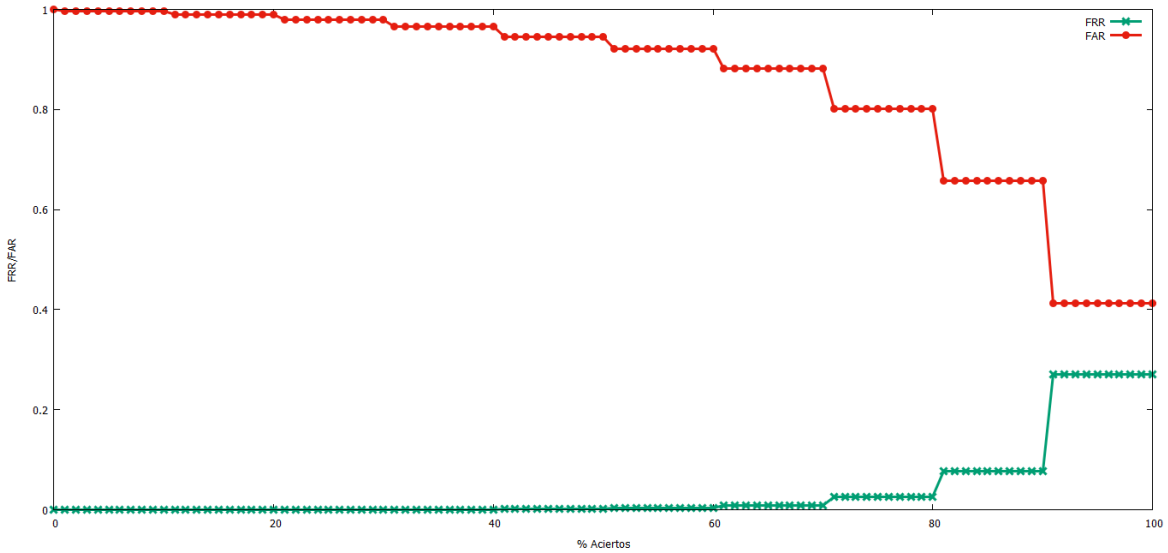
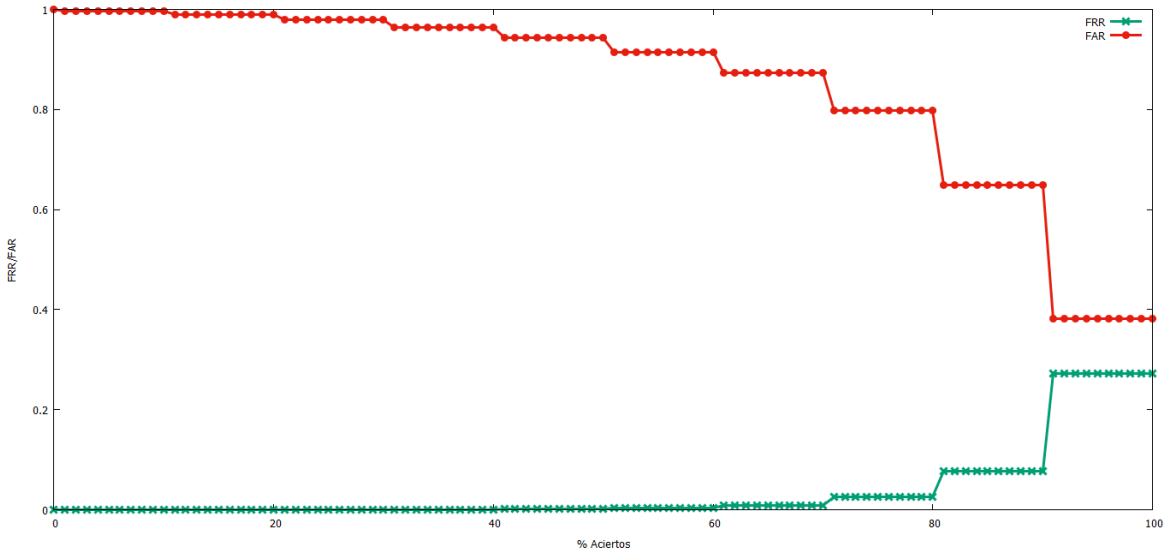



Figura 93: FT4



# Anexo IV: Tabla ERR MAN

| Combinaciones | Umbrales | ERR |
|---|---|---|
| **DT, FT1, FT2, FT3, FT4** | DT(1.75), FT1(2.25), FT2(2.25), FT3(1.5), FT4(1.5) | 0.15180883 |
| **DT, FT1, FT2, FT3** | DT(1.75), FT1(1.5), FT2(2.25), FT3(1.5) | 0.15134314 |
| **DT, FT1, FT2, FT4** | DT(1.75), FT1(2.25), FT2(1.5), FT4(1.5) | 0.14908333 |
| **DT, FT1, FT2** | DT(1.75), FT1(1.5), FT2(1.5) | 0.1489902 |
| **DT, FT1, FT3, FT4** | DT(1.75), FT1(2.25), FT3(1.5), FT4(1.5) | 0.15180883 |
| **DT, FT1, FT3** | DT(1.75), FT1(1.5), FT3(1.5) | 0.15134314 |
| **DT, FT1, FT4** | DT(1.75), FT1(1.5), FT4(1.75) | 0.15027206 |
| **DT, FT1** | DT(1.5), FT1(1.75) | 0.13884804 |
| **DT, FT2, FT3, FT4** | DT(1.75), FT2(2.25), FT3(1.5), FT4(1.5) | 0.15180883 |
| **DT, FT2, FT3** | DT(1.75), FT2(1.5), FT3(1.5) | 0.15211275 |
| **DT, FT2, FT4** | DT(1.75), FT2(1.5), FT4(1.5) | 0.14908579 |
| **DT, FT2** | DT(1.75), FT2(1.5) | 0.15069853 |
| **DT, FT3, FT4** | DT(1.75), FT3(1.5), FT4(1.5) | 0.15180883 |
| **DT, FT3** | DT(1.5), FT3(1.75) | 0.14020097 |
| **DT, FT4** | DT(1.5), FT4(1.75) | 0.13997549 |
| **DT** | DT(1.25) | 0.1463407 |
| **FT1, FT2, FT3, FT4** | FT1(2.25), FT2(1.25), FT3(1.25), FT4(2.0) | 0.21710049 |
| **FT1, FT2, FT3** | FT1(2.0), FT2(1.25), FT3(1.25) | 0.21753922 |
| **FT1, FT2, FT4** | FT1(1.25), FT2(1.5), FT4(1.5) | 0.19787745 |
| **FT1, FT2** | FT1(1.25), FT2(1.25) | 0.2044951 |
| **FT1, FT3, FT4** | FT1(1.25), FT3(1.5), FT4(1.5) | 0.19855148 |
| **FT1, FT3** | FT1(1.25), FT3(1.25) | 0.20969608 |
| **FT1, FT4** | FT1(1.25), FT4(1.5) | 0.1983799 |
| **FT1** | FT1(1.25) | 0.22486764 |
| **FT2, FT3, FT4** | FT2(1.25), FT3(1.25), FT4(2.0) | 0.21723774 |
| **FT2, FT3** | FT2(1.25), FT3(1.25) | 0.2181201 |
| **FT2, FT4** | FT2(1.25), FT4(1.5) | 0.21058333 |
| **FT2** | FT2(1.25) | 0.22151962 |
| **FT3, FT4** | FT3(1.25), FT4(1.25) | 0.20972058 |
| **FT3** | FT3(1.25) | 0.23889461 |
| **FT4** | FT4(1.25) | 0.22279656 |

# Anexo V: Tabla ERR MEANSTDV

| Combinaciones | Umbral | ERR |
|---|---|---|
| **DT, FT1, FT2, FT3, FT4** | 0.35000002 | 0.2227206 |
| **DT, FT1, FT2, FT3** | 0.35000002 | 0.22120589 |
| **DT, FT1, FT2, FT4** | 0.35000002 | 0.21175736 |
| **DT, FT1, FT2** | 0.40000004 | 0.21157354 |
| **DT, FT1, FT3, FT4** | 0.35000002 | 0.21743137 |
| **DT, FT1, FT3** | 0.35000002 | 0.21589951 |
| **DT, FT1, FT4** | 0.40000004 | 0.20553431 |
| **DT, FT1** | 0.45000005 | 0.20033088 |
| **DT, FT2, FT3, FT4** | 0.35000002 | 0.21791667 |
| **DT, FT2, FT3** | 0.35000002 | 0.21286029 |
| **DT, FT2, FT4** | 0.40000004 | 0.20782843 |
| **DT, FT2** | 0.50000006 | 0.1927402 |
| **DT, FT3, FT4** | 0.35000002 | 0.21162745 |
| **DT, FT3** | 0.45000005 | 0.201125 |
| **DT, FT4** | 0.45000005 | 0.19487256 |
| **DT** | 1.0500001 | 0.14603677 |
| **FT1, FT2, FT3, FT4** | 0.25 | 0.23218627 |
| **FT1, FT2, FT3** | 0.25 | 0.23673284 |
| **FT1, FT2, FT4** | 0.3 | 0.23165196 |
| **FT1, FT2** | 0.25 | 0.23287745 |
| **FT1, FT3, FT4** | 0.25 | 0.2329804 |
| **FT1, FT3** | 0.25 | 0.24119608 |
| **FT1, FT4** | 0.3 | 0.22944364 |
| **FT1** | 0.25 | 0.24154165 |
| **FT2, FT3, FT4** | 0.25 | 0.23636764 |
| **FT2, FT3** | 0.25 | 0.24062255 |
| **FT2, FT4** | 0.3 | 0.23281372 |
| **FT2** | 0.3 | 0.23582843 |
| **FT3, FT4** | 0.25 | 0.2395049 |
| **FT3** | 0.2 | 0.24845098 |
| **FT4** | 0.3 | 0.23946078 |

# Anexo VI: Tabla ERR ZSCORE

| Combinaciones | Umbral | ERR |
|---|---|---|
| **DT, FT1, FT2, FT3, FT4** | 93.0 | 0.23301226 |
| **DT, FT1, FT2, FT3** | 91.0 | 0.19470833 |
| **DT, FT1, FT2, FT4** | 91.0 | 0.1922108 |
| **DT, FT1, FT2** | 91.0 | 0.17085785 |
| **DT, FT1, FT3, FT4** | 91.0 | 0.194875 |
| **DT, FT1, FT3** | 91.0 | 0.1733701 |
| **DT, FT1, FT4** | 91.0 | 0.17135294 |
| **DT, FT1** | 91.0 | 0.16303432 |
| **DT, FT2, FT3, FT4** | 91.0 | 0.19585785 |
| **DT, FT2, FT3** | 91.0 | 0.17343381 |
| **DT, FT2, FT4** | 91.0 | 0.17255637 |
| **DT, FT2** | 86.0 | 0.15026961 |
| **DT, FT3, FT4** | 91.0 | 0.17523284 |
| **DT, FT3** | 91.0 | 0.1655 |
| **DT, FT4** | 91.0 | 0.16533089 |
| **DT** | 82.0 | 0.14178921 |
| **FT1, FT2, FT3, FT4** | 98.0 | 0.28227204 |
| **FT1, FT2, FT3** | 97.0 | 0.294125 |
| **FT1, FT2, FT4** | 97.0 | 0.28266913 |
| **FT1, FT2** | 96.0 | 0.29750246 |
| **FT1, FT3, FT4** | 97.0 | 0.2853848 |
| **FT1, FT3** | 96.0 | 0.30674264 |
| **FT1, FT4** | 96.0 | 0.2859853 |
| **FT1** | 91.0 | 0.31766665 |
| **FT2, FT3, FT4** | 97.0 | 0.2962549 |
| **FT2, FT3** | 96.0 | 0.31120098 |
| **FT2, FT4** | 96.0 | 0.29983333 |
| **FT2** | 91.0 | 0.31960538 |
| **FT3, FT4** | 96.0 | 0.31043628 |
| **FT3** | 91.0 | 0.34199265 |
| **FT4** | 91.0 | 0.327375 |